UNIVERSIDAD CATÓLICA SANTA MARÍA

FACULTAD DE CIENCIAS E INGENIERIAS FÍSICAS Y FORMALES

PROGRAMA PROFESIONAL DE INGENIERÍA DE SISTEMAS

"OPTIMIZACION EN LA SEGURIDAD DE ALMACENAMIENTO DE DATOS, UTILIZANDO PROTOCOLOS CRIPTOGRAFICOS MEDIANTE TÉCNICAS GENÉTICAS"

Tesis presentado por el
Bachiller en Ingeniería de
Sistemas:

Herrera Damiani, Jefferson

AREQUIPA – PERÚ

2014

DEDICATORIA

La presente tesis se la dedico a mi familia que gracias a su apoyo pude concluir mi carrera. A mis padres y hermanos por su apoyo y confianza. Gracias por ayudarme a cumplir mis objetivos como persona y estudiante. A mi padre por brindarme los recursos necesarios y estar a mi lado apoyándome y aconsejándome siempre. A mi madre por hacer de mí una mejor persona a través de sus consejos, enseñanzas y amor. A mis hermanos por estar siempre presentes, acompañándome para poderme realizar.

INDICE

GLOSARIO DE TERMINOS

Algoritmo: Conjunto ordenado y finito de operaciones que permite hallar la solución de un problema.

Protocolo: Método estándar que permite la comunicación entre procesos (que potencialmente se ejecutan en diferentes equipos), es decir, es un conjunto de reglas y procedimientos que deben respetarse para el envío y la recepción de datos a través de una red.

Sistema: Conjunto de partes o elementos organizados y relacionados que interactúan entre sí para lograr un objetivo.

Código: Elemento integrante de un sistema comunicativo que le da forma o que cifra al mensaje que pretende ser transmitido.

Software: Conjunto de instrucciones detalladas que controlan la operación de un sistema computacional.

Cifrar: Procedimiento que utiliza un algoritmo de cifrado con cierta clave (clave de cifrado) transforma un mensaje.

Descifrar: Interpretar un mensaje escrito en un lenguaje secreto compuesto por signos especiales utilizando la clave por la cual se ha cifrado el mensaje.

Clave: Código de signos convenidos que se utiliza para transmitir un mensaje secreto o privado.

Datos: Representación simbólica (numérica, alfabética, algorítmica, etc.) de un atributo o variable cuantitativa.

Bit: Dígito del sistema de numeración binario.

Diagrama de Clases: Tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, orientados a objetos.

Diagrama de Casos de Uso: Representa la forma en cómo un Cliente (Actor) opera con el sistema en desarrollo, además de la forma, tipo y orden en como los elementos interactúan.

Diagrama de Estado: Muestra el conjunto de estados por los cuales pasa un objeto durante su vida en una aplicación.

Diagrama de Secuencia: Esquema conceptual que permite representar el comportamiento de un sistema, para lo cual emplea la especificación de los objetos que se encuentran en un escenario y la secuencia de mensajes intercambiados entre ellos.

Diagrama de Bloques: Representación gráfica del funcionamiento interno de un sistema, que se hace mediante bloques y sus relaciones.

Diagrama de Despliegue: Diagrama que se utiliza para modelar el hardware utilizado en las implementaciones de sistemas.

INDICE DE FIGURAS

INDICE DE TABLAS

INDICE DE GRAFICOS

RESUMEN

Los protocolos criptográficos se usan ampliamente para el transporte y el almacenamiento seguros de los datos a nivel de aplicación, utilizando algoritmos criptográficos que son funciones matemáticas usadas en los procesos desencriptación y desencriptación, trabaja en combinación con una llave (un número, palabra, frase, o contraseña) para encriptar y desencriptar datos.

Las técnicas genéticas permiten optimizar la seguridad que brinda los algoritmos criptográficos, ya que pueden mediante acciones aleatorias seleccionar el mensaje cifrado obtenido por el protocolo criptográfico que utilice el emisor y receptor y por medio de selección natural reunirlos y formar un nuevo mensaje cifrado la cual es enviada del emisor al receptor y este último mediante ingeniería inversa recupera el mensaje cifrado original para poder descifrar el mensaje y autentificar si es del emisor.

ABSTRACT

Cryptographic protocols are widely used for safe transportation and storage of data at the application level, using cryptographic algorithms are mathematical functions used in decryption and decryption processes, working in combination with a key (a number, word, phrase, or password) to encrypt and decrypt data.

Genetic techniques to optimize the security provided by the cryptographic algorithms as they can through random actions select the encrypted message obtained by the cryptographic protocol that uses the sender and receiver and through natural selection reunite and form a new encrypted message which is sent from sender to receiver and the latter recovered by reverse engineering the original to authenticate and decrypt the message if the sender encrypted message.

INTRODUCCION

Uno de los grandes inconvenientes de todas las empresas es la seguridad de los datos almacenados de los clientes ya que estos datos pueden ser adquiridos y manipulados por terceros o incluso un intruso malicioso puede logra entrar en el sistema y leer o modificar los datos, hacerse pasar por el cliente, bombardea el sistema con tanto tráfico que éste puede colapsar, intervenir en los medios físicos de la red para capturar información o engañar al sistema.

Tantas son las formas de que nuestros datos o los datos de una empresa puedan ser manipulados por terceros que nos vemos con la necesidad de incrementar la seguridad de las mismas mediante la utilización de protocolos criptográficos, los cuales son utilizados para el cifrado tanto de mensajes como de archivos y es la forma más segura de prevenir la pérdida de información, ya que permiten el intercambio de la información entre un emisor y un receptor por canales inseguros, hay varios protocolo criptográfico que se pueden utilizar como: diffie-hellman, RSA, clave pública, etc. El problema se presenta cuando un hacker conoce el tipo de protocolo que el emisor y receptor utiliza ya que al saberlo puede muy fácilmente aplicar el algoritmo que utilizan y por ende robar la información que se esté intercambiando o almacenando en el caso de una base de datos.

Las Técnicas genéticas se presentan como un método de reforzamiento de la seguridad ya que al utilizarlos con los protocolos criptográficos nos permiten incorporar una capa más de seguridad ya que por más que un hacker sepa el protocolo que los usuarios estén utilizando la información que obtenga seguirá cifrado por los algoritmos genéticos.

CAPITULO I:

PLANTEAMIENTO TEORICO

1.1 OBJETO DE ESTUDIO

Uno de los grandes inconvenientes de todas las empresas es la seguridad de los datos almacenados de los clientes ya que al estar en un ambiente de red de redes es importante y difícil. Es importante pues la información tiene un valor significativo la información puede comprarse y venderse de manera directa o utilizarse directamente para crear productos y servicios nuevos que proporcionan grandes ganancias. La seguridad en una red de redes resulta difícil debido a que implica entender cuándo y cómo pueden confiar los usuarios participantes, las computadoras, los servidores y las redes, uno en otro tratando que la información almacenada no sea adquirida y manipulada por terceros o incluso por un intruso malicioso que puede logra entrar en el sistema y leer o modificar los datos, hacerse pasar por el cliente, bombardear el sistema con tanto tráfico que éste puede colapsar, intervenir en los medios físicos de la red para capturar información o engañar al sistema.

La Criptografía, es una de las disciplinas que se encarga de mantener segura dicha información importante. Con una idea de necesidad actual y técnica que apoye a ser más segura la información significativa, es que se ha realizado el presente trabajo de investigación titulado “OPTIMIZACION EN LA SEGURIDAD DE ALMACENAMIENTO DE DATOS, UTILIZANDO PROTOCOLOS CRIPTOGRAFICOS MEDIANTE TÉCNICAS GENÉTICAS”. El tema de investigación comprende acerca de cómo mediante la utilización de algoritmo genéticos se puede optimizar los protocolos criptográficos lo cual nos brindaría una capas adicional de seguridad para la protección de nuestros datos o de la información que va a ser transmitida, teniendo como indicadores el nivel de seguridad, el tiempo y el tamaño de los datos.

Dicha investigación permite la protección de información significativa para los usuarios ante terceras personas que quieran tener acceso no autorizado a dicha información.

La razón por la que se ha elegido el tema es por la necesidad de contar con una mejora en la seguridad de almacenamiento e intercambio de datos lo que nos permitirá mantener segura la información, esto se puede lograr con la aplicación de técnicas genéticas.

El tema se encuentra ubicado dentro de la encriptación de claves utilizando cualquier tipo de protocolo criptográfico.

Con los conocimientos adquiridos en Criptografía se quiere mejorar la seguridad de información, contando para ello con técnicas genéticas.

La investigación está dirigida a comprobar que es posible mejorar la seguridad de la información utilizando técnicas genéticas.

1.2 OBJETIVO

Optimizar la seguridad de la transmisión y almacenamiento de datos enviados por los usuarios mediante técnicas genéticas.

1.3 OBJETIVO ESPECIFICO

- Evaluar algoritmos criptográficos aplicados en el almacenamiento de datos.
- Evaluar algoritmos genéticos en su desempeño aplicado a la criptografía.
- Comparación del algoritmo desarrollado con los algoritmos existentes de criptografía evaluados para evaluar la reducción huecos de seguridad.

1.4 PROBLEMA

Inseguridad en la transmisión de datos.

1.5 HIPOTESIS

Dado que exististe una falta de seguridad en la transferencia de datos, es probable que utilizando algoritmos criptográficos basados en técnicas genéticas se pueda lograr mejorar seguridad en información enviada.

1.6 VARIABLES

1.6.1 Variable Independiente

Protocolos criptográficos mediante técnicas genéticas

1.6.2 Variable Dependiente

Seguridad en la transferencia y almacenamiento de datos.

1.7 SINTESIS DE VARIABLES E INDICADORES

VARIABLES	INDICADORES
Protocolos criptográficos mediante técnicas genéticas	• Tamaño • Tiempo
Seguridad en la transferencia y almacenamiento de datos	• Nivel de seguridad

1.8 TÉCNICA

La técnica, es el modo de recorrer lo trazado por el método, las técnicas se engloban dentro del método y tienen un carácter operativo y práctico.

Para comprobar si es factible mejorar la seguridad de los datos utilizando técnicas genéticas se optó por realizar un software de encriptación

utilizando cualquier tipo de protocolo criptográfico al que se le implementara técnicas genéticas.

Se definió el sistema de encriptación y desencriptación a utilizar.
Se definió que la forma de encriptarlos mensajes sea codificación en binario.
Se definió la fuente a utilizar para el texto original es decir aquella fuente que genera palabra o letras, ya que el cifrado del mensaje con la clave se lleva a cabo carácter por carácter.
Se definió que al mensaje cifrado se le aplique algoritmo genético
Se definió que el algoritmo genético se repita una vez para comprobar su fiabilidad.
Se definió que solo se utilicé los métodos de selección, combinación y mutación de los algoritmos genéticos.
Se definió que la forma de desencriptar sea primero aplicando algoritmo genético para obtener el código binario original y después utilizar el sistema de desencriptado elegido al principio.

Para la evaluación se utilizó fichas técnicas de observación una encuesta para categorizar la seguridad y establecer la confiabilidad del software.
Así mismo se utilizó la técnica de la observación para determinar la rapidez de encriptamiento y desencriptamiento que demora el software.

1.9 CAMPO DE VERIFICACIÓN

- **Sistemas de Validación de Claves para accesos a Bases de Datos:** SQL, Oracle.
- **Laboratorio de Pruebas para la Comparación de Desempeño:** Computadora Pentium III Procesador Intel Core 2 Duo.
- **Sistema Operativo:** Windows XP.

1.10 METODOLOGIA DE TRABAJO

Se eligió el Modelo Incremental por que combina elementos del Modelo Lineal Secuencial con la filosofía interactiva de Construcción de Prototipos. Cada secuencia lineal produce un incremento del software.

En una visión genérica, el proceso se divide en 4 partes:

- Análisis
- Diseño
- Código
- Prueba

Durante el proceso se trata de llevar a cabo al proyecto en diferentes partes que al final terminará siendo la solución completa requerida por el cliente, pero éstas partes no se pueden realizar en cualquier orden, sino que dependen de lo que el cliente este necesitando con más urgencia, de los puntos más importantes del proyecto, los requerimientos más básicos, difíciles y con mayor grado de riesgo, ya que estos se deben hacer al comienzo, de manera que se disminuya la dificultad y el riesgo en cada versión.

De este modo podemos terminar una aplicación ejecutable (primera versión) que podrá ser entregada al cliente para que éste pueda trabajar en ella y el programador pueda considerar las recomendaciones que el cliente efectúe para hacer mejoras en el producto. Estas nuevas mejoras deberán esperar a ser integradas en la siguiente versión junto con los demás requerimientos que no fueron tomados en cuenta en la versión anterior.

1.11 DE LA INVESTIGACION

1.11.1 Tipo de Investigación

Investigación aplicada ya que se desarrolla en la experimentación, validación y comparación sobre protocolos criptográficos.

1.11.2 Nivel de Investigación

Explicativo porque se pretende proponer una optimización en la seguridad de almacenamiento de los datos.

1.11.3 Área de Investigación

- Inteligencia Artificial

1.11.4 Línea de Investigación

- Criptografía
- Algoritmos Genéticos

1.12 METODO Y DISEÑO DE LA INVESTIGACION

1.12.1 Método de la investigación

Método científico

1.12.2 Diseño de la investigación

Para la investigación se utilizan las siguientes técnicas, instrumentos y materiales de verificación.

- Técnicas: Observación
- Variables: Protocolos Criptográfico
- Instrumentos: Fichas de observación

1.13 COBERTURA DEL ESTUDIO

1.13.1 Universo

Usuarios expertos

1.13.2 Muestra

No probabilístico

1.14 TECNICAS E INSTRUMENTOS DE RECOLECCION DE INFORMACION

1.14.1 Técnicas

- Revisión Bibliográfica
- Documentación

1.14.2 Instrumentos

- Entrevista
- Cuestionario.

1.15 ESTADO DEL ARTE

Para brindar una mejor seguridad de los datos ya sea almacenándolos o intercambiándolos es que se utiliza métodos o sistemas de encriptación los cuales nos ayudan a incrementar la seguridad tal como lo podemos ver en el trabajo de "*D. M., G. (1993). Designing and Detecting Trapdoors for Discrete Log. Crytosystems Advances in Crytology. Springer Verlag*".y tambien en el trabajo de "*E. Newton, D. (01 de october de 1997). Encyclopedia of Crytology. ABC-CLIO".* los cuales mediante la utilizacion de funciones algebraicas como se indica en el libro de"*D. Lipson, J. (1981).*

Elements of Algebra and Algebraic Computing. Addison-Wesley" permiten la codificacion y decodificacion de los mensajes enviados.Los algoritmos con la implementación de estas funciones algebraicas como se ve en el libro escrito por "*W, D., & Hellman, M. (1976). New Directions in cryptography, IEEE Transactions on Information Theory."*Son utilizados para brindar seguridad a los mensajes enviados, también se utilizan protocolos los cuales usan a estos algoritmos para dar seguridad a la red tal como se ve en "*Gutiérrez Gutierrez, J. (2003). Protocolos criptograficos y seguridad en redes. universidad de cantabria*". Aunque estos algoritmos tienden a sufrir vulnerabilidades o ataces según el trabajo de "*Lucena Lopez, M. J. (Junio de 2010). Criptografia y Seguridad en Computadoras*".

Por tal motivo se ha optado por demostrar que los algoritmos genéticos pueden ser una forma más óptima para brindar una mejor seguridad a los datos transferidos y almacenados, como se aprecia en el trabajo de "*Melanie, M. (1999). An introduccion to Genetic Algorithms. MIT Press*". obteniendo resultados interesantes en la practica como lo demuestra en trabajo de "*L. Haup, R., & Ellen Haup, S. (2004). Practical Genetic Algorithms. Wiley*".Al final la importancia de la seguridad de los datos es crucial ya que no solo para las empresas sino también para todas las personas la información resulta ser un factor primordial en nuestras vidas como se ve en el trabajo de "*Ramio Aguirre, J. (marzo de 2006). Seguridad Informatica. Universidad Politecnica de Madrid . Cuarta Edicion v32.*"

CAPITULO II:

MARCO TEORICO

2.1 SEGURIDAD INFORMÁTICA

Toda organización debe estar a la vanguardia de los procesos de cambio. Donde disponer de información continua, confiable y en tiempo, constituye una ventaja fundamental. Donde tener información es tener poder.

La seguridad informática, es el área de la informática que se enfoca en la protección de la infraestructura computacional y todo lo relacionado con ésta (incluyendo la información contenida). Para ello existen una serie de estándares, protocolos, métodos, reglas, herramientas y leyes concebidas para minimizar los posibles riesgos a la infraestructura o a la información(Ramio Aguirre, 2006). La seguridad informática comprende software, bases de datos, metadatos, archivos y todo lo que la organización valore (activo) y signifique un riesgo si ésta llega a manos de otras personas. Este tipo de información se conoce como información privilegiada o confidencial.

El concepto de seguridad de la información no debe ser confundido con el de seguridad informática, ya que este último sólo se encarga de la

seguridad en el medio informático, pero la información puede encontrarse en diferentes medios o formas, y no solo en medios informáticos.

La seguridad informática es la disciplina que se ocupa de diseñar las normas, procedimientos, métodos y técnicas destinados a conseguir un sistema de información seguro y confiable.(Gutierrez Gutierrez, 2003)

La seguridad informática debe garantizar:

- La Disponibilidad de los sistemas de información.
- El Recupero rápido y completo de los sistemas de información
- La Integridad de la información.
- La Confidencialidad de la información.

2.2 CRIPTOLOGÍA

Desde que el hombre ha necesitado comunicarse con los demás ha tenido la necesidad de que algunos de sus mensajes solo fueran conocidos por las personas a quien estaban destinados. La necesidad de poder enviar mensajes de forma que sólo fueran entendidos por los destinatarios hizo que se crearan sistemas de cifrado (criptografía), de forma que un mensaje después de un proceso de transformación, lo que llamamos cifrado, solo pudiera ser leído siguiendo un proceso de descifrado.

La criptología tiene por objeto esconder la información contenida en ciertos datos, así como transformarla por motivos como identificación del propietario de los datos, del autor, del remitente de un mensaje, compresión, etc. La criptografía es la rama de la criptología que se encarga de encriptar y desencriptar, o lo que es lo mismo, cifrar y descifrar mensajes(Borisov, Golberg, & Brewer, 2004). Encriptación o cifrado es el acto de convertir el texto original en texto encriptado o cifrado por medio de un algoritmo y un secreto de encriptación, el texto cifrado debe ser ininteligible para alguien que no posea el secreto de desencriptación. Análogamente, la desencriptación nos devuelve el texto original a partir del

texto cifrado por medio de un algoritmo y un secreto de desencriptación. El criptoanálisis es la rama que se encarga de obtener el texto original a partir del texto cifrado, sin poseer acceso legítimo al secreto de encriptación.(Newton, 1997)

La criptografía clásica cuenta con métodos muy variados, sin embargo, al no contar con criptoanálisis sistemático, estos métodos resultan hoy día bastante vulnerables, es decir, cualquiera con un presupuesto razonable puede hacerse de los recursos humanos y de cómputo necesarios para romper los métodos clásicos esencialmente en tiempo real. Por lo tanto, estos métodos son no sólo inservibles, sino incluso peligrosos, pues al ser utilizados por individuos ignorantes y en productos vendidos por compañías igualmente ignorantes o sin escrúpulos, brindan una ilusión de seguridad que es peor que la franca ausencia de la misma, pues en tal caso, al menos se está consciente de la vulnerabilidad, con lo cual uno evita usar el canal de comunicaciones inseguro para transmitir datos sensibles.

En contraste, la criptografía moderna puede proteger incluso a individuos de escasos recursos, no sólo del familiar o colega entrometido, sino también de gobiernos poderosos o grandes corporaciones multinacionales, es decir, de enemigos con vastos recursos que pueden, incluso, incluir laboratorios de criptoanálisis. Para esto, la criptografía moderna echa mano de algoritmos públicos revisados por pares implementados en programas de cómputo o software. Este punto merece aclararse y enfatizarse: un algoritmo secreto es peligroso, pues al no haber sido revisado por la comunidad académica, constituye un hueco potencial en la seguridad del usuario.(Gutierrez Gutierrez, 2003) Así, se afirma que a menos que éste tenga un departamento de investigación en criptología, un algoritmo secreto es una muy mala idea. Cabe mencionar, que incluso un experto puede cometer errores triviales y en el caso en el que nadie revise su trabajo, no habrá quien descubra esa situación. Por esto, los algoritmos de

encriptación modernos tienen dos tipos de entradas: el texto a ser encriptado y la o las llaves de encriptación. De esta forma, el algoritmo se hace público para que pueda ser revisado y el secreto, que sigue siendo necesario, se transfiere a las llaves de encriptación y /o desencriptación.

2.2.1 PRINCIPIOS DE ENCRIPTACIÓN

Existen dos tipos principales de algoritmos de encriptación sobre los cuales están construidos los protocolos más complicados: encriptación simétrica y asimétrica, también conocidas respectivamente como de llave secreta y de llave pública. Los primeros algoritmos modernos que se desarrollaron son los simétricos. La simetría a la que alude el nombre se da entre las llaves, es decir, estos algoritmos usan la misma llave para encriptar y para desencriptar. Esto tiene el inconveniente de que para establecer un canal de comunicación seguro con estos algoritmos, el emisor y el receptor se tienen que poner de acuerdo en una llave.(Cipher A., Kahn, Kruh, & Mellen, 1987) De estar lejos el uno del otro, su seguridad se vería comprometida si un enemigo interceptara la transmisión de la llave, lo cual se conoce como un ataque de capa y espada. Otros ataques de capa y espada a los que es susceptible la criptografía de llave simétrica incluyen soborno, chantaje, intimidación y tortura del emisor o receptor o incluso, de algún Courier o mensajero que se use para la llave. Estos ataques permitirían al enemigo obtener todos los datos transmitidos. Sin embargo, hay ataques incluso más peligrosos, por ejemplo el conocido como man-in-the-middle u hombre-en medio. En este ataque el enemigo consigue interrumpir el canal de comunicaciones, por ejemplo, cortando un cable y uniendo ambos extremos a un equipo de cómputo o por ejemplo, interceptando a un Courier. Así, cuando la llave es enviada, el enemigo la recibe y la sustituye por

otra. Es evidente que este ataque compromete la seguridad del canal de formas mucho más severas, pues el interceptor puede no sólo escuchar los datos transmitidos, sino también sustituirlos por otros fabricados por él. En fin, la criptografía simétrica implica este problema de distribución de llaves, por eso, también se conoce como criptografía de llave secreta.(Borisov, Golberg, & Brewer, 2004)

Es aquí donde entra la criptografía asimétrica o de llave pública. En este caso, cualquiera que vaya a actuar como receptor, genera una pareja de llaves, mediante un algoritmo accesorio, que se conocen como llave pública y llave privada. A continuación, reparte su llave pública a cualquier posible emisor, incluso, posiblemente publicándola en algún medio masivo o base de datos y guarda celosamente su llave privada. De esta forma, cualquiera que quiera mandarle un mensaje, lo encripta con su llave pública y se evita el problema de distribución de llaves, pues la llave pública no sirve para desencriptar los datos cifrados con ella misma.

En la práctica, la encriptación de llave pública es mucho más pesada computacionalmente que la simétrica; por esta razón, a menudo se usan esquemas híbridos donde un algoritmo asimétrico se utiliza para establecer un canal seguro, y este canal se emplea para distribuir la llave para un algoritmo simétrico, a esta llave se le conoce como llave de sesión, pues para mayor seguridad se actualizará en cada nueva sesión de comunicación. Así se obtiene lo mejor de dos mundos: se evita el problema de distribución de llaves, pero se aprovecha el bajo costo computacional del algoritmo simétrico durante el resto de la sesión. Hay algunos esquemas híbridos incluso más complicados, los cuales están más allá del enfoque de este trabajo. (Gordon, 1993)

Además de permitir el intercambio seguro de mensajes, la criptología usa los esquemas y algoritmos de encriptación para elaborar protocolos complejos que permiten, por ejemplo: repartir un secreto entre m personas de tal manera que se necesiten, al menos, n para abrirlo; crear dinero digital anónimo, de tal manera que un banco emisor pueda comprobar su autenticidad y evitar su duplicidad sin conocer la identidad del usuario; así como para comprobarle a alguien con certeza casi absoluta que se conoce un secreto sin divulgar el mismo.

2.2.2 CRIPTOGRAFÍA

Tradicionalmente se ha definido como la parte de la criptología que se ocupa de las técnicas, bien sea aplicadas al arte o la ciencia, que alteran las representaciones lingüísticas de mensajes, mediante técnicas de cifrado y/o codificado, para hacerlos ininteligibles a intrusos (lectores no autorizados) que intercepten esos mensajes. Por tanto el único objetivo de la criptografía era conseguir la confidencialidad de los mensajes. Para ello se diseñaban sistemas de cifrado y códigos(Gutierrez Gutierrez, 2003). En esos tiempos la única criptografía que había era la llamada criptografía clásica.

La aparición de la Informática y el uso masivo de las comunicaciones digitales han producido un número creciente de problemas de seguridad. Las transacciones que se realizan a través de la red pueden ser interceptadas. La seguridad de esta información debe garantizarse. Este desafío ha generalizado los objetivos de la criptografía para ser la parte de la criptología que se encarga del estudio de los algoritmos, protocolos (se les llama protocolos criptográficos) y sistemas que se utilizan para proteger la información

y dotar de seguridad a las comunicaciones y a las entidades que se comunican.

Para ello los criptógrafos investigan, desarrollan y aprovechan técnicas matemáticas que les sirven como herramientas para conseguir sus objetivos. Los grandes avances que se han producido en el mundo de la criptografía han sido posibles gracias a los grandes avances que se ha producido en el campo de las matemáticas y la informática.

La criptografía actualmente se encarga del estudio de los algoritmos, protocolos y sistemas que se utilizan para dotar de seguridad a las comunicaciones, a la información y a las entidades que se comunican. El objetivo de la criptografía es diseñar, implementar, implantar, y hacer uso de sistemas criptográficos para dotar de alguna forma de seguridad. Por tanto el tipo de propiedades de las que se ocupa la criptografía son por ejemplo:(Borisov, Golberg, & Brewer, 2004)

- Confidencialidad. Es decir garantiza que la información está accesible únicamente a personal autorizado. Para conseguirlo utiliza códigos y técnicas de cifrado.
- Integridad. Es decir garantiza la corrección y completitud de la información. Para conseguirlo puede usar por ejemplo funciones hash criptográficas MDC, protocolos de compromiso de bit, o protocolos de notarización electrónica.
- No repudio. Es decir proporcionar protección frente a que alguna de las entidades implicadas en la comunicación, pueda negar haber participado en toda o parte de la comunicación. Para conseguirlo se puede usar por ejemplo firma digital. En algunos contextos lo que se intenta es justo lo contrario:

Poder negar que se ha intervenido en la comunicación. Por ejemplo cuando se usa un servicio de mensajería instantánea y no queremos que se pueda demostrar esa comunicación. Para ello se usan técnicas como el cifrado negable.

- Autenticación. Es decir proporciona mecanismos que permiten verificar la identidad del comunicante. Para conseguirlo puede usar por ejemplo función hash criptográfica MAC o protocolo de conocimiento cero.
- Soluciones a problemas de la falta de simultaneidad en la tele firma digital de contratos. Para conseguirlo puede usar por ejemplo protocolos de transferencia inconsciente.

Un sistema criptográfico es seguro respecto a una tarea si un adversario con capacidades especiales no puede romper esa seguridad, es decir, el atacante no puede realizar esa tarea específica.

Según(Gutierrez Gutierrez, 2003) la criptografía es: “La rama inicial de las Matemáticas y en la actualidad de la Informática y la Telemática, que hace uso de métodos y técnicas con el objeto principal de cifrar y/o proteger un mensaje o archivo por medio de un algoritmo, usando una o más claves. Esto da lugar a diferentes tipos de sistemas de cifra que permiten asegurar estos cuatro aspectos de la seguridad informática: la confidencialidad, la integridad, la disponibilidad y el no repudio de emisor y receptor.”

Que como podemos pensar es una descripción mucho más acertada que la que nos podemos encontrar en muchos libros, así como la definición que nos hace la *Real Academia de la Lengua RAE.*

Otra definición a tener en cuenta es el significado de *criptoanálisis,* el cual es el arte y la ciencia de transgredir las claves de acceso, es decir la que se encarga de descifrar los mensajes.(Borisov, Golberg, & Brewer, 2004)

2.2.2.1 LA CRIPTOGRAFÍA MODERNA

La criptografía moderna se basa en las mismas ideas básicas que la criptografía tradicional, la transposición y la sustitución, pero con distinta orientación. En la criptografía moderna el objetivo es hacer algoritmos de cifrado complicados y rebuscados.

Según el tratamiento del mensaje se dividen en:

2.2.2.1.1 CIFRADO EN BLOQUE

DES (Data Encryption Standard)

- El texto original se codifica en bloques de 64 bits, clave de 56 bits y 19 etapas diferentes.
- El descifrado se realiza con la misma clave y los pasos inversos.
- El inconveniente es que puede ser descifrado probando todas las combinaciones posibles, cosa que queda solucionada con Doble DES (ejecuta el DES 2 veces con 3 claves distintas) y el Triple Des (2 claves y 3 etapas).(Lucena Lopez, 2010)

IDEA (International Data Encryption Algorithm)

- Tenemos una clave de 128 bits con 8 iteraciones.
- El descifrado se realiza aplicando el mismo algoritmo pero con subclaves diferentes. (Lucena Lopez, 2010)

RSA (Rivest, Shamir y Adleman)

Se basa en la dificultad de factorizar número grandes por parte de los ordenadores.

Los pasos son:

- Seleccionar 2 números primos grandes.
- Calcular n=p*q y z= (p-1)*(q-1).
- Seleccionar un número primo d con respecto a z
- Encontrar e tal que e*d = 1 (mod z)→e*d mod z = 1.
- El algoritmo es el siguiente:
- Dividimos el texto normal en bloques P, que cumplen que 0<P<n
- Para cifrar un mensaje P calculamos C = p^e(mod n).
- Para descifrar C calculamos P = C^d (mod n).

El principal inconveniente como es de suponer es la lentitud. Hay que destacar que el RSA pertenece a los algoritmos con clave pública mientras que el DES y el IDEA son algoritmos de clave secreta.(Lucena Lopez, 2010)

2.2.2.1.2 Cifrado en flujo (A5, RC4, SEAL) cifrado bit a bit

A5 Es el algoritmo de cifrado de voz. Gracias a él, la conversación va encriptado. Se trata de un algoritmo de flujo (streamcipher) con una clave de 64 bits. Hay dos versiones, denominadas A5/1 y A5/2; esta última es la versión autorizada para la exportación, y en consecuencia resulta más fácil de atacar.(Lucena Lopez, 2010)

Según el tipo de claves se dividen en:

2.2.2.1.2.1 Cifrado con clave secreta o Criptosistemas simétricos

Existirá una única clave (secreta) que deben compartir emisor y receptor. Con la misma clave se cifra y se descifra por lo que la seguridad reside sólo en mantener dicha clave en secreto.

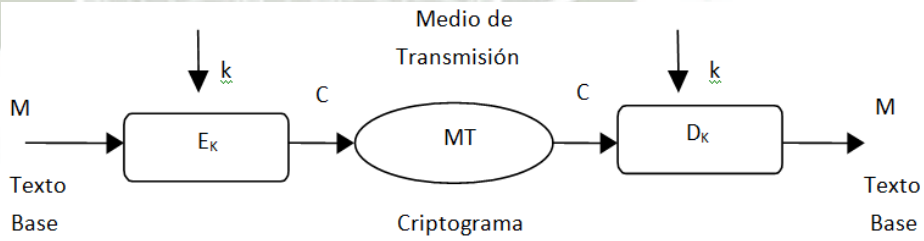

Figura 2.1: Diagrama de Criptosistema Base

(Gordon, 1993)

Con Ek ciframos el mensaje original aplicándole la clave k y con Dk lo desciframos, aplicándole de la misma forma la clave k.

La confidencialidad y la integridad se lograrán si se protegen las claves en el cifrado y en el descifrado. Es decir, se obtienen simultáneamente si se protege la clave secreta.(Gordon, 1993)

2.2.2.1.2.2 Cifrado con clave pública o Criptosistemas asimétricos

Cada usuario crea un par de claves, una privada para descifrar y otra pública para cifrar, inversas dentro de un cuerpo finito. Lo que se cifra en emisión con una clave, se descifra en recepción con la clave inversa. La seguridad del sistema reside en la dificultad computacional de descubrir la clave privada a partir de la pública. Para ello, usan funciones matemáticas de un solo sentido con trampa.

El nacimiento de la criptografía asimétrica se dio al estar buscando un modo más práctico de intercambiar las claves simétricas. Diffie y Hellman(Diffie & Hellman, 1976), proponen una forma para hacer esto, sin embargo no fue hasta que el popular método de Rivest Shamir y Adleman RSA publicado en 1978, cuando toma forma la criptografía asimétrica, su funcionamiento está basado en la imposibilidad computacional de factorizar números enteros grandes.(Newton, 1997)

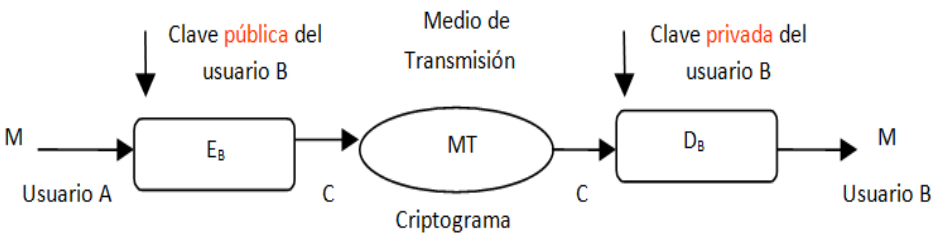

Figura 2.2 Diagrama de Criptosistema de Clave Pública

(Newton, 1997)

Hay que tener en cuenta que Eb y Db son inversas dentro de un cuerpo, además se debe de tener en cuenta que se cifra con la clave pública del destinatario, de forma que conseguimos que solo él, al tener su clave privada pueda acceder al mensaje original.

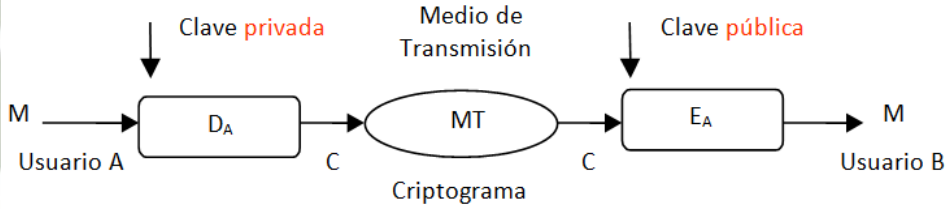

Figura 2.3 : Diagrama de Criptosistema de Clave Privada

(Lucena Lopez, 2010)

En este segundo caso podemos observar cómo está basado en el cifrado con la clave privada del emisor y al igual que antes hay que tener en cuenta que Ea y Da son inversas dentro de un cuerpo.(Lucena Lopez, 2010)

Llegados a este punto la pregunta que nos deberíamos de hacer es, que utilizar, clave pública o privada, pues bien, como siempre depende:

- Los sistemas de clave pública son más lentos, aunque como hemos visto es posible que no sean tan seguros. Hay algunos tipos de ataques que les pueden afectar.

- Los sistemas de clave privada son más lentos, aunque son más seguros, los algoritmos son más complejos y es más difícil su traducción por otros sujetos.

2.2.2.1.3 Sistemas CEE (de curvas elípticas).

CCE es otro tipo de criptografía de clave pública es el que usa curvas elípticas definidas en un campo finito. La diferencia que existe entre este sistema y RSA es el problema matemático en el cual basan su seguridad. RSA razona de la siguiente manera: te doy el número 15 y te reta a encontrar los factores primos. El problema en el cual están basados los sistemas que usan curvas elípticas que denotaremos como CCE es el del logaritmo discreto elíptico, en este caso su razonamiento con números sería algo como: te doy el número 15 y el 3 y te reta a encontrar cuantas veces tienes que sumar el mismo 3 para obtener 15.(Lucena Lopez, 2010)

Los CCE basan su seguridad en el Problema del Logaritmo Discreto Elíptico (PLDE), esto quiere decir que dados P, Q puntos de la curva hay que encontrar un número entero x tal

que xP = Q (xP = P+P+…+P, x veces). Obsérvese que a diferencia del PFE (Problema de Factorización Entera) el PLDE no maneja completamente números, lo que hace más complicado su solución.(Lipson, 1981)

La creación de un protocolo con criptografía de curvas elípticas requiere fundamentalmente una alta seguridad y una buena implementación, para el primer punto se requiere que la elección de la curva sea adecuada, principalmente que sea no-supe singular y que el orden del grupo de puntos racionales tenga un factor primo de al menos 163 bits, además de que este orden no divida al orden de un número adecuado de extensiones del campo finito, para que no pueda ser sumergido en él, si el campo es ZP, se pide que la curva no sea anómala o sea que no tenga p puntos racionales. Todo esto con el fin de evitar los ataques conocidos.

Para el caso de la implementación hay que contar con buenos programas que realicen la aritmética del campo finito, además de buenos algoritmos que sumen puntos racionales, se toma una base polinomial que tenga el mínimo de términos por ejemplo un trinomio para generar los elementos del campo finito esto si la implementación es en software, y se toma una base normal si es en hardware. Además de contemplar que las operaciones de puntos racionales pueden hacerse en el espacio proyectivo, esto elimina el hacer divisiones, ahorrando tiempo.

Otra noticia sobre CCE es que los elementos de los puntos racionales pueden ser elementos de un campo finito de característica 2, es decir pueden ser arreglos de ceros y unos de longitud finita (01001101110010010111), en este

caso es posible construir una aritmética que optimice la rapidez y construir un circuito especial para esa aritmética, a esto se le conoce como Base Normal Optima.

Lo anterior permite con mucho que los CCE sean idóneos para ser implementados en donde el poder de cómputo y el espacio del circuito sea reducido, donde sea requerida una alta velocidad de procesamiento o grandes volúmenes de transacciones, donde el espacio de almacenamiento, la memoria o el ancho de banda sea limitado. Lo que permite su uso en Smart Cards, Teléfonos celulares, Fax, Organizadores de Palma, PCs, etcétera.(Lucena Lopez, 2010)

En la actualidad existen varios estándares que permiten el uso adecuado y óptimo de los CCE, entre los cuales se encuentran: IEEE P1363 (Institute of Electrical and ElectronicsEngineers), el ANSI X9.62, ANSI X9.63, ANSI TG-17, ANSI X12 (American NationalStandardsInstitute), UN/EDIFACT, ISO/IEC 14888, ISO/IEC 9796-4, ISO/IEC 14946 (International StandardsOrganization), ATM Forum (AsynchronousTransportMode), WAP (WirelessApplicationProtocol). En comercioelectrónico: FSTC (Financial Services Technology Consortion), OTP 0.9 (Open Trading Protocol), SET (Secure Electronic Transactions). En internet IETF (The Internet Engineering Task Force), IPSec (Internet Protocol Security Protocol).(Gutierrez Gutierrez, 2003)

2.2.2.2 EL NO REPUDIO

El no repudio, consiste en que el receptor puede saber a ciencia cierta de quien es el mensaje y esto lo podemos conseguir mediante la firma digital, al ser esta única, como si fuera una firma normal en un papel, tenemos como un acuse o un recibo, que demuestra quién ha enviado el mensaje.

Esto es un añadido a todo lo visto anteriormente que incluye más seguridad a la transmisión de mensajes entre los usuarios. Además la firma digital puede ser utilizada al igual que el hash tanto en los sistemas de clave pública como en los de clave privada. Por este motivo es muy utilizada en documentos legales y financieros.(Ramio Aguirre, 2006)

Requisitos de la firma digital:

- Debe ser fácil de generar.
- Será irrevocable, no rechazable por su propietario.
- Será única, sólo posible de generar por su propietario.
- Será fácil de autenticar o reconocer por su propietario y los usuarios receptores.
- Debe depender del mensaje y del autor

Un pequeño ejemplo:

- Tenemos una clave Pública (nA, eA) y una clave Privada (dA)
- Firma :rAh(M) = h(M)dAmodnA
- A envía el mensaje M en claro (o cifrado) al destinatario B junto a la firma: {M, rAh(M)}

- El destinatario B tiene la clave pública eA,nA de A y descifra rAh(M) ⇒ {(h(M)dA)eAmodnA} obteniendo así h(M).

- Como recibe el mensaje M', calcula la función hash h(M') compara:→ Si h(M') = h(M) se acepta la firma.

2.2.3 CRIPTOANÁLISIS

El criptoanálisis es la ciencia opuesta a la criptografía (quizás no es muy afortunado hablar de ciencias *opuestas*, sino más bien de ciencias *complementarias*), ya que si ésta trata principalmente de crear y analizar criptosistemas seguros, la primera intenta romper esos sistemas, demostrando su vulnerabilidad: dicho de otra forma, trata de descifrar los criptogramas(Lucena Lopez, 2010). El término *descifrar* siempre va acompañado de discusiones de carácter técnico, aunque asumiremos que descifrar es conseguir el texto en claro a partir de un criptograma, sin entrar en polémicas de reversibilidad y solidez de criptosistemas.

En el análisis para establecer las posibles debilidades de un sistema de cifrado, se han de asumir las denominadas condiciones del peor caso: (1) el criptoanalista tiene acceso completo al algoritmo de encriptación, (2) el criptoanalista tiene una cantidad considerable de texto cifrado, y (3) el criptoanalista conoce el texto en claro de parte de ese texto cifrado. También se asume generalmente el Principio de Kerckhoffs (Newton, 1997), que establece que la seguridad del cifrado ha de residir exclusivamente en el secreto de la clave, y no en el mecanismo de cifrado.

Aunque para validar la robustez de un criptosistema normalmente se suponen todas las condiciones del peor caso, existen ataques más

específicos, en los que no se cumplen todas estas condiciones. Cuando el método de ataque consiste simplemente en probar todas y cada una de las posibles claves del espacio de claves hasta encontrar la correcta, nos encontramos ante un ataque de fuerza bruta o ataque exhaustivo. Si el atacante conoce el algoritmo de cifrado y sólo tiene acceso al criptograma, se plantea un ataque sólo al criptograma; un caso más favorable para el criptoanalista se produce cuando el ataque cumple todas las condiciones del peor caso; en este caso, el criptoanálisis se denomina de texto en claro conocido. Si además el atacante puede cifrar una cantidad indeterminada de texto en claro al ataque se le denomina de texto en claro escogido; este es el caso habitual de los ataques contra el sistema de verificación de usuarios utilizado por Unix, donde un intruso consigue la tabla de contraseñas (generalmente /etc/passwd) y se limita a realizar cifrados de textos en claro de su elección y a comparar los resultados con las claves cifradas (a este ataque también se le llama de diccionario, debido a que el atacante suele utilizar un fichero `diccionario' con los textos en claro que va a utilizar). El caso más favorable para un analista se produce cuando puede obtener el texto en claro correspondiente a criptogramas de su elección; en este caso el ataque se denomina de texto cifrado escogido. Cualquier algoritmo de cifrado, para ser considerado seguro, ha de soportar todos estos ataques y otros no citados; sin embargo, en la criptografía, como en cualquier aspecto de la seguridad, informática o no, no debemos olvidar un factor muy importante: las personas. El sistema más robusto caerá fácilmente si se tortura al emisor o al receptor hasta que desvelen el contenido del mensaje, o si se le ofrece a uno de ellos una gran cantidad de dinero; este tipo de ataques (sobornos, amenazas, extorsión, tortura) se consideran siempre los más efectivos.

2.2.3.1 CRIPTOANÁLISIS Y ATAQUES A CRIPTOSISTEMAS

El criptoanálisis es el arte de descifrar comunicaciones encriptadas sin conocer las llaves correctas. Existen muchas técnicas criptoanalíticas. Algunas de las más importantes se describen a continuación.

- **Ataques a textos cifrados (Ciphertext-onlyattack)**

  Esta es la situación en la cual el atacante no conoce nada sobre el contenido del mensaje, y debe trabajar solo desde el texto cifrado. En la práctica es muy probable hacer tantas conjeturas acerca del texto plano, como cantidad de tipos de mensajes tengan un encabezado similar. Incluso las cartas y los documentos ordinarios comienzan de una manera muy previsible(Lucena Lopez, 2010). Por ejemplo, muchos ataques clásicos utilizan "análisis frecuencial" del texto cifrado, sin embargo, no funciona bien contra los cifrados modernos. Los criptosistemas modernos no son débiles contra ataques de "texto cifrado", aunque algunas veces son considerados con el agregado de que el mensaje contiene "tendencia" estática.

- **Ataques de texto plano conocidos**

  El atacante conoce o puede adivinar el texto de alguna parte del texto cifrado. La tarea es desencriptar el resto del bloque cifrado utilizando esta información. Esto puede ser hecho determinando la clave utilizada para encriptar la información, o a través de algún atajo.(Gordon, 1993)

Uno de los mejores ataques modernos de texto plano conocido es el "criptoanálisis lineal" contra cifradores de bloques.

- **Ataques de texto plano seleccionado**

  El atacante puede tener cualquier texto encriptado con una llave desconocida. La tarea es determinar la llave utilizada para encriptar. Un buen ejemplo de este ataque es el "criptoanálisis diferencial" que puede ser aplicado a cifradores de bloques y, en algunos casos, a funciones Hash. Algunos criptosistemas, particularmente el RSA, son vulnerables a estos ataques. Cuando tales algoritmos son utilizados, se debe tener cuidado en el diseño de la aplicación (o protocolo) de forma tal que un atacante no pueda obtener el texto encriptado.(Gordon, 1993)

- **Ataque de hombre en medio**

  Este ataque es relevante para las comunicaciones criptográficas y los protocolos de intercambio de llaves. La idea es que cuando dos partes, A y B, están intercambiando llaves por comunicaciones seguras (por ejemplo utilizando Diffie-Hellman), un adversario (intruso) se posiciona entre A y B en la línea de comunicación. El intruso intercepta las señales que A y B se envían, y ejecuta un intercambio de llaves entre A y B. A y B terminaran utilizando llaves diferentes, cada una de las cuales es conocida por el intruso. El intruso puede luego desencriptar cualquier comunicación de A con la llave que comparte con A, y luego reenviarla a B encriptándola nuevamente con la llave que comparte con B. Ambos A y B

pensarán que se están comunicando en forma segura pero de hecho el intruso está escuchando todo.(Gordon, 1993)

La forma habitual de prevenir este ataque es utilizar un sistema de clave pública capaz de proveer firmas digitales. Por configuración, las partes deben conocer de antemano la clave pública de cada una de ellas. Después de que han sido generadas, las partes se envían firmas digitales. El hombre de por medio falla en el ataque a causa de que no es capaz de falsificar las firmas sin conocer las llaves privadas utilizadas para generar las firmas.

Esta solución es suficiente si existe también una manera segura de distribuir claves públicas. Una forma es la jerarquía de certificados como X.509. Es utilizado por ejemplo en IPSec.

- La **correlación** entre la clave secreta y la salida del criptosistema es la fuente principal de información para el criptoanálisis. En el caso más simple, la información sobre la llave secreta es filtrada por el criptosistema. Casos más complicados requieren estudios sobre la correlación (básicamente, cualquier relación que no sería esperada) entre la información observada(o tomada) de los criptosistemas y la información de la llave adivinada.(Gordon, 1993)

Por ejemplo, en ataques lineales contra bloques cifrados el criptoanálisis estudia el texto plano conocido y el observado. Adivinando algunos bits del criptosistema el analista determina, por correlación entre el texto plano y el cifrado, si el "adivinó bien". Esto se puede repetir y tiene muchas variantes. El criptoanálisis diferencial introducido

por Eli Biham y Adi Shamir en los '80 fue el primer ataque que utilizó completamente esta idea contra los bloques cifrados (especialmente contra el DES). Más tarde Eli Biham y Adi Shamir introducen el criptoanálisis lineal que fue aún más efectivo contra el DES. Más recientemente, se han desarrollado nuevos ataques que utilizan ideas similares.(Cipher A., Kahn, Kruh, & Mellen, 1987)

La idea correlacional es fundamental para la criptografía y muchas investigaciones han tratado de construir criptosistemas que sean seguros contra tales ataques.

La idea de la correlación es fundamental para la criptografía y varias investigaciones han tratado de construir criptosistemas que han demostrado ser seguros contra tales ataques.

- **Ataques contra el hardware o utilizando el hardware base**

  Así como en los últimos años más y más dispositivos pequeños de criptografía han sido ampliamente utilizados, una nueva categoría de ataques se ha hecho relevante la cual apunta directamente a las implementaciones de hardware de los criptosistemas.

  Los ataques utilizan datos muy buenos obtenidos del dispositivo criptográfico, supongamos, información de la encriptación y de la llave de estas medidas. Las ideas básicas están estrechamente relacionadas con aquellos en otros ataques correlacionados. Por ejemplo, el atacante adivina algunos bit de la clave y trata de verificar la exactitud de lo adivinado estudiando la correlación contra lo que el adivinó.

Se han propuesto varios ataques como la utilización cuidadosa del cronometraje del dispositivo, medidas del consumo de energía, y patrones de radiación. Estas mediciones pueden ser utilizadas para obtener la llave secreta u otro tipo de información almacenada en el dispositivo. Estos ataques son independientes de los algoritmos criptográficos utilizados y pueden ser aplicados a cualquier dispositivo que no esté explícitamente protegido.

- **Las Fallas en los Criptosistemas:** Pueden conducir al criptoanálisis y aún al descubrimiento de la llave secreta. El interés en dispositivos criptográficos conduce al descubrimiento de que algunos algoritmos se comportan muy mal con la introducción de una pequeña falla en el cálculo interno. (Lucena Lopez, 2010)

  Por ejemplo, la implementación usual de una operación de llave privada RSA es susceptible a los ataques de fallas. Se ha sido demostrado que causando un bit de error en un punto adecuado puede revelar la factorización del módulo (revela la llave privada).

  Se han aplicado ideas similares a una gran variedad de algoritmos y dispositivos. Es así necesario que los dispositivos criptográficos sean diseñados para ser altamente resistentes a fallas (y contra introducciones maliciosas de fallas por criptoanálisis).

Informática (o cálculo) Cuántico: escritos de Peter Shor sobre factoreo polinómico del tiempo y algoritmos logarítmicos discretos con informática cuántica han causado el creciente interés en la informática cuántica. La informática cuántica es un campo reciente de investigación que utiliza mecanismos cuánticos para construir

computadoras que son, en teoría, más potentes que las modernas computadoras seriales.(Lucena Lopez, 2010) El poder es derivado del paralelismo inherente de los mecanismos cuánticos. Así que en lugar de hacer una tarea a la vez, como lo hacen los mecanismos seriales, las computadoras cuánticas pueden ejecutarlas todas al mismo tiempo. De esa manera se espera que con las computadoras cuánticas podamos resolver problemas que no son resueltos por los seriales. Los resultados de Shor implican o sugieren que si las computadoras cuánticas pueden ser implementadas eficientemente entonces muchas de las llaves públicas criptográficas serán historia. Sin embargo, son mucho menos efectivas contra llaves criptográficas secretas.

El estado de arte actual de las computadoras cuánticas no aparenta ser alarmante, ya que solo se han implementado máquinas muy pequeñas. La teoría de la informática cuántica brinda más promesas en cuanto al rendimiento que las computadoras seriales, sin embargo, que se realice en la práctica es una cuestión pendiente.

2.2.4 Criptografía DNA

Leonard Adleman (uno de los inventores del RSA) trajo a colación la idea de utilizar DNA como computadoras. Las moléculas de DNA pueden ser vistas como una gran computadora capaz de realizar ejecuciones en paralelo.(Haupt & Haupt, 2004) Esta naturaleza concurrente podría brindar a las computadoras DNA incrementos exponenciales de velocidad contra las computadoras seriales modernos.

Desafortunadamente hay problemas con las computadoras DNA, el crecimiento exponencial de la velocidad implica también la necesidad de crecimiento del volumen de material requerido. De esa manera las computadoras DNA tienen limitaciones de rendimiento en la práctica. Además, no es muy fácil construir una.

2.2.5 CRIPTOSISTEMA

Un Criptosistema se define como la quíntupla (m,C,K,E,D), donde:

- m representa el conjunto de todos los mensajes sin cifrar (texto plano) que pueden ser enviados.
- C Representa el conjunto de todos los posibles mensajes cifrados, o criptogramas.
- K representa el conjunto de claves que se pueden emplear en el Criptosistema.
- E es el conjunto de transformaciones de cifrado o familia de funciones que se aplica a cada elemento de m para obtener un elemento de C. Existe una transformación diferente E_k para cada valor posible de la clave K.
- D es el conjunto de transformaciones de descifrado, análogo a E.

Todo Criptosistema cumple la condición $D_k(E_k(m))=m$ es decir, que si se tiene un mensaje m, se cifra empleando la clave K y luego se descifra empleando la misma clave, se obtiene el mensaje original m." (Gordon, 1993)

Existen dos tipos fundamentales de Criptosistemas utilizados para cifrar datos e información digital y ser enviados posteriormente después por medios de transmisión libre.

a. Simétricos o de clave privada: se emplea la misma clave K para cifrar y descifrar, por lo tanto el emisor y el receptor deben poseer la clave. El mayor inconveniente que presentan es que se debe contar con un canal seguro para la transmisión de dicha clave.
b. Asimétricos o de llave pública: se emplea una doble clave conocidas como K_p (clave privada) y K_P (clave Pública). Una de ellas es utilizada para la transformación E de cifrado y la otra para el descifrado D. En muchos de los sistemas existentes estas clave son intercambiables, es decir que si empleamos una para cifrar se utiliza la otra para descifrar y viceversa.

Los sistemas asimétricos deben cumplir con la condición que la clave Pública (al ser conocida y sólo utilizada para cifrar) no debe permitir calcular la privada. Como puede observarse este sistema permite intercambiar claves en un canal inseguro de transmisión ya que lo único que se envía es la clave pública.

Los algoritmos asimétricos emplean claves de longitud mayor a los simétricos. Así, por ejemplo, suele considerarse segura una clave de 128 bits para estos últimos pero se recomienda claves de 1024 bits (como mínimo) para los algoritmos asimétricos(Newton, 1997). Esto permite que los algoritmos simétricos sean considerablemente más rápidos que los asimétricos.

En la práctica actualmente se emplea una combinación de ambos sistemas ya que los asimétricos son computacionalmente más costosos (mayor tiempo de cifrado). Para realizar dicha combinación se cifra el mensaje m con un sistema simétrico y luego se encripta la

clave K utilizada en el algoritmo simétrico (generalmente más corta que el mensaje) con un sistema asimétrico(Lucena Lopez, 2010).

Después de estos Criptosistemas modernos podemos encontrar otros no menos importantes utilizados desde siempre para cifrar mensajes de menos importancia o domésticos, y que han ido perdiendo su eficacia por ser fácilmente criptoanalizables y por tanto "reventables". Cada uno de los algoritmos clásicos descriptos a continuación utiliza la misma clave K para cifrar y descifrar el mensaje.

En la siguiente figura podemos observar un ejemplo de un criptosistema que nos muestra como sería el funcionamiento esquemático, sea cual sea el canal de transmisión, del cifrado y descifrado de un mensaje en su paso del transmisor al receptor.

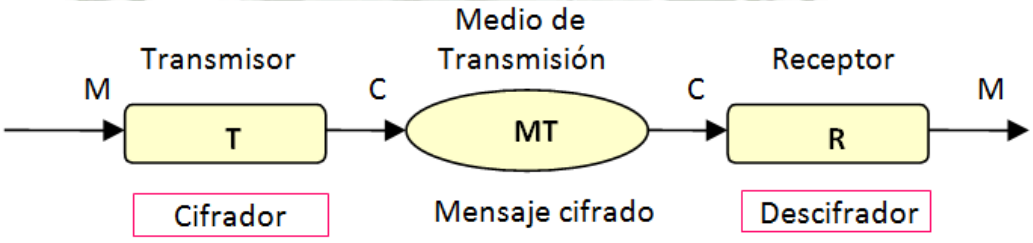

Figura 2.4: Ejemplo de Criptosistemas

(Fuente Propia)

2.2.5.1 TRANSPOSICIÓN

Son aquellos que alteran el orden de los caracteres dentro del mensaje a cifrar. El algoritmo de transposición más común consiste en colocar el texto en una tabla de n columnas. El texto cifrado serán los caracteres dados por columna (de arriba hacia

abajo) con una clave K consistente en el orden en que se leen las columnas.

Ejemplo: Si n = 3 columnas, la clave K es (3,1,2) y el mensaje a cifrar "SEGURIDAD INFORMATICA".

1	2	3
S	E	G
U	R	I
D	A	D
	I	N
F	O	R
M	A	T
I	C	A

El mensaje cifrado será: " GIDNRTASUD FMIERAIOAC "

2.2.5.2 CIFRADOS MONOALFABÉTICOS

Sin desordenar los símbolos del lenguaje, se establece una correspondencia única para todos ellos en todo el mensaje. Es decir que si al carácter A le corresponde carácter D, esta correspondencia se mantiene durante todo el mensaje.

- **Algoritmo de César**

  Es uno de los algoritmos criptográficos más simples. Consiste en sumar 3 al número de orden de cada letra. De

esta forma a la A le corresponde la D, a la B la E, y así sucesivamente. Puede observarse que este algoritmo ni siquiera posee clave, puesto que la transformación siempre es la misma. Obviamente, para descifrar basta con restar 3 al número de orden de las letras del criptograma.

Ejemplo: Si el algoritmo de cifrado es:

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
D E F G H I J K L M N O P Q R S T U V W X Y Z A B C

Entonces el mensaje cifrado será:

S E G U R I D A D I N F O R M A T I C A

V H J X U L G D G L Q I R U P D W L F D

- **Sustitución General**

  Es el caso general del algoritmo de César. El sistema consiste en sustituir cada letra por otra aleatoria. Esto supone un grado más de complejidad aunque como es de suponer las propiedades estadísticas del texto original se conservan en el criptograma y por lo tanto el sistema sigue siendo criptoanalizable.

2.3 PROTOCOLO CRIPTOGRAFICO

Un protocolo de seguridad (también llamado protocolo criptográfico o protocolo de cifrado) es un protocolo abstracto o concreto que realiza funciones relacionadas con la seguridad, aplicando métodos criptográficos.
Los protocolos criptográficos son algoritmos distribuidos que constan de una secuencia de pasos o etapas que tienen que ser realizados por dos o

más entidades para alcanzar unos determinados objetivos de seguridad. (Gutierrez Gutierrez, 2003)

Un protocolo describe la forma en que un algoritmo debe usarse. Un protocolo lo suficientemente detallado incluye detalles acerca de las estructuras de datos y representaciones, punto en el cual puede usarse para implementar versiones interoperables múltiples de un programa.

Los protocolos criptográficos se usan ampliamente para transporte de datos seguros a nivel de aplicación. Un protocolo criptográfico comúnmente incorpora por lo menos uno de los siguientes aspectos:

- Establecimiento de claves
- Autenticación de entidades
- Cifrado simétrico y autenticación de mensajes
- Transporte de datos en forma segura a nivel de aplicación
- Métodos de no repudio

Por ejemplo, TransportLayer Security (TLS) es un protocolo criptográfico usado en conexiones web (HTTP) seguras(Gutierrez Gutierrez, 2003). Posee un mecanismo de autenticación de entidades basado en el sistema X.509, una fase de configuración de claves, en la cual se decide una clave de cifrado simétrico mediante el uso de criptografía de clave pública, y una función de transporte de datos de nivel de aplicación. Estos tres aspectos tienen interconexiones importantes. El TLS estándar no provee apoyo para no repudio.

Hay otros tipos de protocolos criptográficos también e incluso el término mismo tiene varias interpretaciones distintas. Los protocolos criptográficos de *aplicación* usan a menudo uno o más métodos de acuerdo de claves, a los cuales a veces se los llama "protocolos criptográficos". De hecho, el TLS emplea el intercambio de claves de Diffie-Hellman(Diffie & Hellman, 1976), el cual si bien no forma parte del TLS, puede ser visto como un protocolo criptográfico por sí mismo para otras aplicaciones.

Los protocolos criptográficos pueden ser verificados formalmente en un nivel abstracto algunas veces.

2.3.1 ALGORITMOS CRIPTOGRAFICOS A EVALUAR

A continuación se indicara el funcionamiento de los algoritmos a evaluar así como sus vulnerabilidades.

2.3.1.1 DIFFIE-HELLMAN

El protocolo criptográfico Diffie-Hellman, debido a Whitfield Diffie y Martin Hellman, (Diffie–HellmanProblem->DHP) es un protocolo de establecimiento de claves entre partes que no han tenido contacto previo, utilizando un canal inseguro, y de manera anónima (no autentificada).(Diffie & Hellman, 1976)

Se emplea generalmente como medio para acordar claves simétricas que serán empleadas para el cifrado de una sesión (establecer clave de sesión). Siendo no autenticado, sin embargo, provee las bases para varios protocolos autenticados.

Su seguridad radica en la extrema dificultad (conjeturada, no demostrada) de calcular logaritmos discretos en un cuerpo finito.

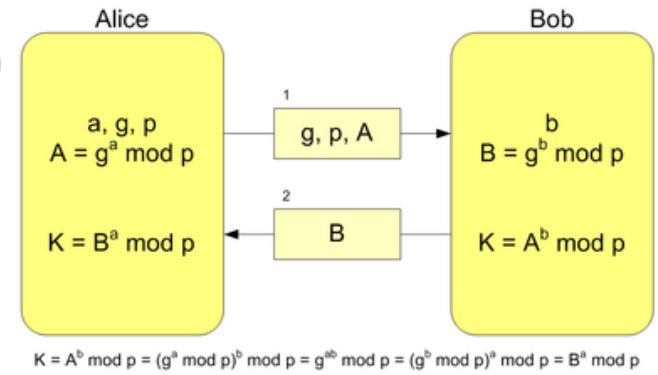

Figura 2.5: Algoritmo Diffie-Hellman

(Diffie & Hellman, 1976)

Para dos partes *Alice* y *Bob* que intentan establecer una clave secreta y un adversario *Mallory*, la versión básica es como sigue:

- Se establecen un primo py un generador $g \in \mathbf{Z}_p^*$(2). Estos son públicos, conocidos no sólo por las partes *Alice* y *Bob* sino también por el adversario *Mallory*
- *Alice* escoge $a \in \mathbf{Z}_{p-1}$al azar, calcula $A = g^a \bmod p$, y envía Aa *Bob*
- *Bob* escoge $b \in \mathbf{Z}_{p-1}$al azar, calcula $B = g^b \bmod p$, y envía Ba *Alice*

Nótese que tanto A como B pueden calcular el valor $K = g^{a \cdot b} \bmod p$. En efecto, lo podemos demostrar usando las propiedades del grupo $\mathbf{Z}_p^*$:

Para Alice:

$$B^a \bmod p = (g^b \bmod p)^a \bmod p = (\overbrace{(g^b \bmod p)(g^b \bmod p)\cdots(g^b \bmod p)}^{a}) \bmod p = g^{b \cdot a} \bmod p = g^{a \cdot b} \bmod p = K$$

Para Bob:

$$A^b \bmod p = (g^a \bmod p)^b \bmod p = (\overbrace{(g^a \bmod p)(g^a \bmod p)\cdots(g^a \bmod p)}^{b}) \bmod p = g^{a \cdot b} \bmod p = K$$

Como ambas partes pueden calcular Kentonces la podemos usar como clave compartida.

Para corregir la vulnerabilidad del protocolo, éste debe ser utilizado conjuntamente con algún sistema que autentique los mensajes. Esto ocurre, por ejemplo, durante el establecimiento de la asociación HIP, donde los paquetes R1 e I2, además de

contener los mensajes de Diffie-Hellman, están firmados digitalmente.

En la figura siguiente se muestra un ejemplo de funcionamiento del protocolo Diffie-Hellman.

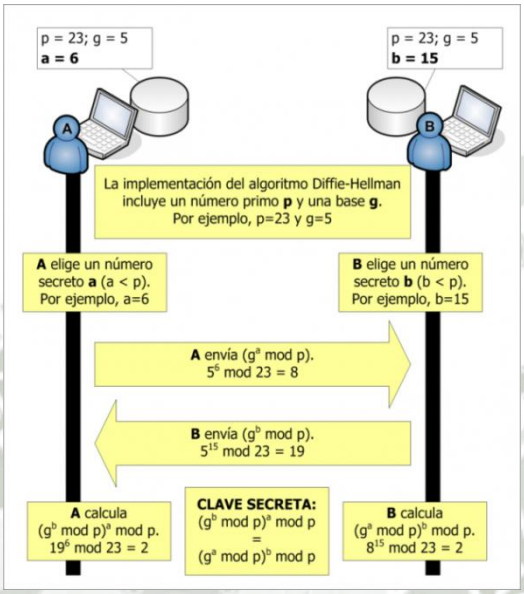

Figura 2.6: Ejemplo de Funcionamiento del Protocolo Diffie-Hellman

(Diffie & Hellman, 1976)

Los valores de “p” y “g” son públicos y cualquier atacante puede conocerlos, pero esto no supone una vulnerabilidad. Aunque un atacante conociese dichos valores y capturara los dos mensajes enviados entre las máquinas A y B, no sería capaz de averiguar la clave secreta. A continuación se muestra la información capturada por un atacante en el escenario de la Figura 2.6:

$(g^a \bmod p) = 8 \rightarrow (5^a \bmod 23) = 8$

$(g^b \bmod p) = 19 \rightarrow (5^b \bmod 23) = 19$

A partir de las ecuaciones anteriores, intentar calcular los valores de “a” y “b” es lo que se conoce como el problema del algoritmo discreto, un problema que se cree computacionalmente intratable y cuya notación es la siguiente:

$a = \log disc_g (g^a \bmod p) = \log disc\ _5 (8)$

$b = \log disc_g (g^b \bmod p) = \log disc\ _5 (19)$

Con los valores del ejemplo sí que es posible encontrar la solución, ya que se ha escogido un número primo “p” muy pequeño (p = 23), y se sabe que “a” y “b” son menores que “p”. Por lo tanto, para obtener los valores secretos en este ejemplo, un atacante tendría que probar sólo 22 posibles valores.

Por suerte, las implementaciones actuales del protocolo Diffie-Hellman utilizan números primos muy grandes, lo que impide a un atacante calcular los valores de “a” y “b”. El valor “g” no necesita ser grande, y en la práctica su valor es 2 ó 5. En el RFC 3526 aparecen publicados los números primos que deben utilizarse. A modo de ejemplo, se facilita aquí el número primo de 1024 bytes propuesto. El valor “g” utilizado es 2:

$p = 2^{8192} – 2^{8128} – 1 + 2^{64} \times ((2^{8062} \text{ pi}) + 4743158)$

Ataques al Algoritmo Diffie-Hellman

Se pueden clasificar en dos grupos: ataque pasivos y ataques activos.

Un ataque pasivo es aquel en el que el “espía” trata de descifrar algo a partir de información cifrada interceptada. Un ataque activo es aquel en el atacante desea no sólo espiar información

interceptada, sino también poder manipularla a su conveniencia.(Gordon, 1993)

Sería un típico ataque pasivo intentar obtener a partir de p, g, y_a, y_b la clave secreta z_{ab}. No obstante, es difícil que este ataque pueda ser llevado a término, ya que para ello se necesitaría obtener x_a o x_b, y para ello debería resolverse alguna de las operaciones:

$x_a = \log_g y_a \pmod p$

$x_b = \log_g y_b \pmod p$

Lo que es inviable para números grandes, así, bastaría elegir p y g lo suficientemente grandes para evitar este ataque.

Así las cosas, para realizar un ataque activo un buen método sería que el atacante (sea llamado C) interviniese de forma activa en el intercambio. Supóngase que C genera un entero aleatorio x_c / $1<x_c<p-1$. Así, cuando A envíe a B y_a, C intercepta la comunicación y envía a B $y_c = g^{x_c} \pmod p$.(Lucena Lopez, 2010)

B recibe y_c , creyendo que procede de A, y responde enviando y_b. Nuevamente, C intercepta la comunicación, y ahora envía y_c a A.

Así, A calcula: $z_{ca} = y^{x_a}{}_c = g^{x_c x_a} \pmod p$

Y B calcula: $z_{cb} = y^{x_b}{}_c = g^{x_c x_b} \pmod p$

Ambas claves que también pueden ser calculadas por el atacante. Así, cuando A envíe una información a B cifrada con z_{ca}, el atacante la intercepta, decodifica, manipula a su antojo, la encripta con z_{cb} y la envía a B. De igual forma cuando B envía información cifrada a A.

Este ataque es difícil de evitar y de descubrir, pero requiere la continua intervención del atacante para no ser descubierto.

2.3.1.2 RSA

RSA (Rivest, Shamir y Adleman) es un sistema criptográfico de clave pública desarrollado en 1977. Es el primer y más utilizado algoritmo de este tipo y es válido tanto para cifrar como para firmar digitalmente.

Este algoritmo es de tipo asimétrico por bloques siendo uno de los más sencillos de comprender e implementar, se basa en la dificultad para factorizar números grandes ya que tanto la clave pública como privada se generan del producto de dos primos grandes.(Lucena Lopez, 2010)

Supongamos que Bob quiere enviar a Alicia un mensaje secreto que solo ella pueda leer.

Alicia envía a Bob una caja con una cerradura abierta, de la que solo Alicia tiene la llave. Bob recibe la caja, escribe el mensaje, lo pone en la caja y la cierra con su cerradura (ahora Bob no puede leer el mensaje). Bob envía la caja a Alicia y ella la abre con su llave. En este ejemplo, la caja con la cerradura es la «clave pública» de Alicia, y la llave de la cerradura es su «clave privada».

Técnicamente, Bob envía a Alicia un «mensaje llano» M en forma de un número $m$menor que otro número$n$, mediante un protocolo reversible conocido como *paddingscheme*(«patrón de relleno»). A continuación genera el «mensaje cifrado» c mediante la siguiente operación:

$$c \equiv m^e \pmod{n},$$

Donde e es la clave pública de Alicia.

Ahora Alicia descifra el mensaje en clave c mediante la operación inversa dada por

$$m \equiv c^d \pmod{n},$$

Donde d es la clave privada que solo Alicia conoce

Para generar la clave pública y la privada se realiza lo siguiente:

1. Cada usuario elige dos números primos distintos p y q.
    - Por motivos de seguridad, estos números deben escogerse de forma aleatoria y deben tener una longitud en bits parecida. Se pueden hallar primos fácilmente mediante test de primalidad.
2. Se calcula $n = pq$.
    - n se usa como el módulo para ambas claves, pública y privada.
3. Se calcula $\varphi(n) = (p-1)(q-1)$, donde φ es la función φ de Euler.
    - Se escoge un entero positivo e menor que $\varphi(n)$, que sea coprimo con $\varphi(n)$.
4. e se da a conocer como el exponente de la clave pública.
    - Si se escoge un e con una suma encadenada corta, el cifrado será más efectivo. Un exponente e muy pequeño (p. ej. $e = 3$) podría suponer un riesgo para la seguridad.

5. Se determina un d(mediante aritmética modular) que satisfaga la congruencia$e \cdot d \equiv 1 \pmod{\varphi(N)}$, es decir, que d sea el multiplicador modular inverso de $e \mod \varphi(n)$

   - Expresado de otra manera,$de - 1$es dividido exactamente por$\varphi(n) = (p-1)(q-1)$.
   - Esto suele calcularse mediante el algoritmo de Euclides extendido(Lipson, 1981).
   - $d$se guarda como el exponente de la clave privada.

La clave pública es(n, e), esto es, el módulo y el exponente de cifrado. La clave privada es(n, d), esto es, el módulo y el exponente de descifrado, que debe mantenerse en secreto.

En el momento de cifrar el mensaje Alicia comunica su clave pública(n, e) a Bob y guarda la clave privada en secreto. Ahora Bob desea enviar un mensaje Ma Alicia.

Primero, Bob convierte$M$en un número entero$m$menor que n mediante un protocolo reversible acordado de antemano. Luego calcula el texto cifrado$c$mediante la operación

$$c \equiv m^e \pmod{n}.$$

Esto puede hacerse rápido mediante el método de exponenciación binaria. Ahora Bob transmite ca Alicia.

Cuando se quiera descifrar el mensaje Alicia puede recuperar$m$a partir de$c$ usando su exponente $d$de la clave privada mediante el siguiente cálculo:

$$m \equiv c^d \pmod{n}.$$

Ahora que tiene m en su poder, puede recuperar el mensaje original M invirtiendo el *paddingscheme*.

El procedimiento anterior funciona porque

$$c^d = (m^e)^d \equiv m^{ed} \pmod{n}.$$

Esto es así porque, como hemos elegido d y e de forma que $ed = 1 + k\varphi(n)$, se cumple

$$m^{ed} = m^{1+k\varphi(n)} = m(m^{\varphi(n)})^k \equiv m \pmod{n}.$$

La última congruencia se sigue directamente del teorema de Euler(Lipson, 1981)cuando m es coprimo con n. Puede demostrarse que las ecuaciones se cumplen para todo m usando congruencias y el teorema chino del resto.

Esto muestra que se obtiene el mensaje original:

$$m \equiv c^d \pmod{n}.$$

Ataques al Algoritmo RSA

Ataque basado en un exponente publico bajo. Aunque un exponente bajo acelera el, cifrado también lo hace más inseguro. Si se encripta e(e+1)/2 mensajes linealmente dependientes con diferentes claves públicas y el mismo valor de e. Si los mensajes son idénticos entonces es suficiente con e mensajes para poder saber el contenido real.

Ataque a modulo común. Un error que se da a la hora de usar RSA es asignar el mismo modulo n a varios usuarios y solo asignar diferentes valores para los exponentes e y d, de tal forma que el texto en claro puede recuperarse sin ninguno de los exponentes privados, como se verá a continuación. Sea P el texto

en claro. Sean e_1 y e_2 los exponentes públicos (claves de cifrado) y n el modulo. Los textos cifrados son:

$$C_1 = P^{e_1} \mod n$$
$$C_2 = P^{e_2} \mod n$$

El atacante tiene acceso a C_1,C_2, e_1, e_2 y n, así para recuperar P y ser e_1y $e_2$primos entre sí, existen r y s tales que $r*e_1+ s*e_2=1$. Supongamos sin pérdida de generalidad que r es negativo (alguno de los dos ha de serlo) entonces puede calcularse el inverso de C_1 y se tendrá que aplicar lo siguiente:

$$(C_1^{-1})^r \cdot C_2^s = P \mod n$$

Ataque Cíclico. Teniendo conocimiento de la clave pública, la utilizara para repetir cuantas veces sea necesario el proceso de cifrado, hasta obtener el mismo valor de bit cifrado, al conseguirlo, el valor del proceso anterior corresponde al valor real de ese bit.(Lucena Lopez, 2010)

2.3.1.3 AES

AdvancedEncryption Standard (AES), también conocido como Rijndael (pronunciado "Rain Doll" en inglés), es un esquema de cifrado por bloques adoptado como un estándar de cifrado por el gobierno de los Estados Unidos. El AES fue anunciado por el Instituto Nacional de Estándares y Tecnología (NIST) como FIPS PUB 197 de los Estados Unidos (FIPS 197) el 26 de noviembre de 2001 después de un proceso de estandarización que duró 5 años. Se transformó en un estándar efectivo el 26 de mayo de 2002. Desde 2006, el AES es uno de los algoritmos más populares usados en criptografía simétrica.(Lucena Lopez, 2010)

Estrictamente hablando, AES no es precisamente Rijndael (aunque en la práctica se los llama de manera indistinta) ya que Rijndael permite un mayor rango de tamaño de bloques y longitud de claves; AES tiene un tamaño de bloque fijo de 128 bits y tamaños de llave de 128, 192 o 256 bits, mientras que Rijndael puede ser especificado por una clave que sea múltiplo de 32 bits, con un mínimo de 128 bits y un máximo de 256 bits.

La mayoría de los cálculos del algoritmo AES se hacen en un campo finito determinado.

AES opera en una matriz de 4×4 bytes, llamada state (algunas versiones de Rijndael con un tamaño de bloque mayor tienen columnas adicionales en el state).

Pseudocódigo

Claves:

Por ser simétrico, se utiliza la misma clave para encriptar como para desencriptar, la longitud de la clave puede ser de 128, 192 o 256 bits según especifica el estándar.

Partiendo de una clave inicial de 16 bytes (128 bits), que también se la puede ver como un bloque o matriz de 4x4 bytes, se generan 10 claves, estas claves resultantes junto con la clave inicial son denominadas subclaves. El proceso de generación de subclaves parte de la clave inicial vista como una matriz de 4x4 bytes.

Para calcular la primera columna de la siguiente subclave se toma la última columna de la subclave anterior (en este caso la clave inicial) y se aplica una operación llamada Rotword que consiste en realizar una rotación del primer byte hacia el último lugar en la columna.

Luego, a la columna resultante, se aplica una operación llamada SubBytes que consiste en reemplazar cada byte de la columna ya rotada por un byte almacenado en una tabla llamada S-Box, esta tabla contiene pre calculados el resultado de aplicarle a cada byte la inversión en el campo GF y una transformación afín, la dimensión de la tabla es de 16x16 bytes donde los índices tanto de las columnas como de las filas van de 0 a F, para obtener la transformación S-Box de un byte se toman los primeros 4 bits como el índice de la fila de la tabla y los segundos 4 como índice de la columna de la tabla.

Luego al resultado se le aplica un XOR byte a byte con la columna 4 posiciones atrás (en este caso la primer columna de la clave inicial) y un XOR byte a byte con una columna de una tabla llamada RCON que mantiene en la primer fila constantes 2i en el campo GF y en las restantes filas 0, por ser la primer subclave la que estamos calculando se toma para el cálculo la primer columna de la tabla RCON.

El resultado de esta última operación será la primera columna de la subclave calculada, Para calcular las tres columnas siguientes se hace un XOR entre la columna anterior y la columna de cuatro posiciones atrás. Una vez aplicadas estas operaciones se tiene una nueva subclaveY se procede de la misma forma para calcular las siguientes subclaves.

Al finalizar se tendrán 11 subclaves, cada una de estas subclaves se aplica en una de las rondas de operaciones que se explican en detalle a continuación.

Etapa inicial:

- AddRoundKey

  Rondas:

- SubBytes — en este paso se realiza una sustitución no lineal donde cada byte es reemplazado con otro de acuerdo a una tabla de búsqueda.

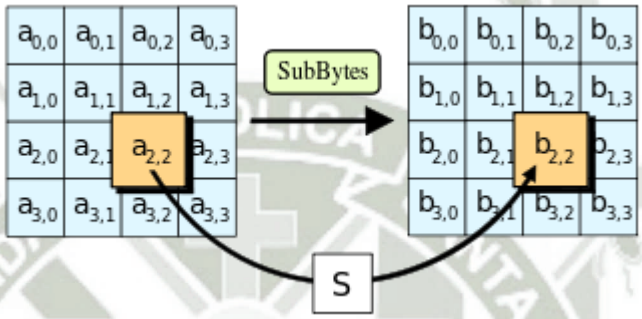

Figura 2.7: Diagrama de SubBytes

(Lucena Lopez, 2010)

- ShiftRows — en este paso se realiza una transposición donde cada fila del «state» es rotada de manera cíclica un número determinado de veces.

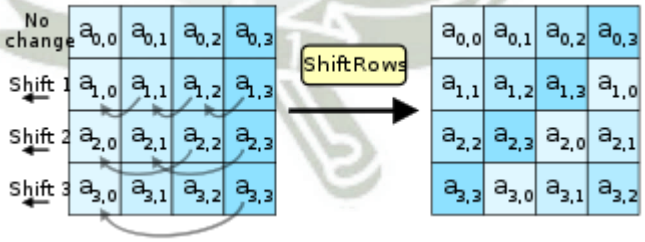

Figura 2.8: Diagrama de ShiftRows

(Lucena Lopez, 2010)

- MixColumns — operación de mezclado que opera en las columnas del «state», combinando los cuatro bytes en cada columna usando una transformación lineal.

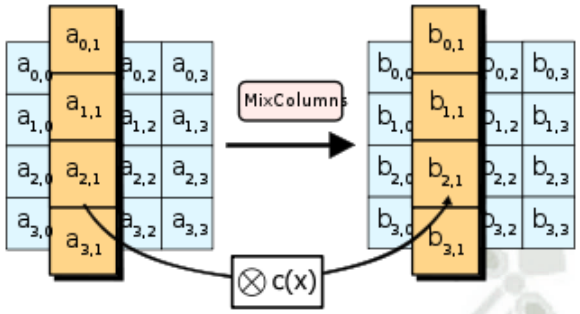

Figura 2.9: Diagrama de MixColumns

(Lucena Lopez, 2010)

- AddRoundKey — cada byte del «state» es combinado con la clave «round»; cada clave «round» se deriva de la clave de cifrado usando una iteración de la clave.

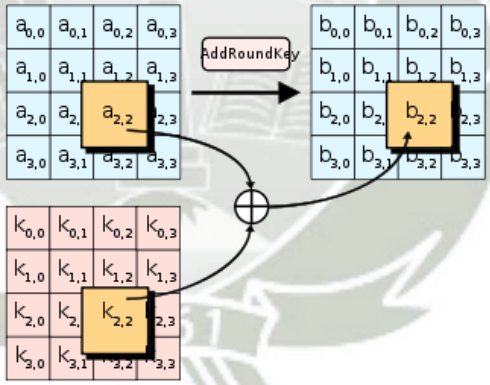

Figura 2.10: Diagrama de AddRoundKey

(Lucena Lopez, 2010)

Etapa final:

- SubBytes
- ShiftRows
- AddRoundKey

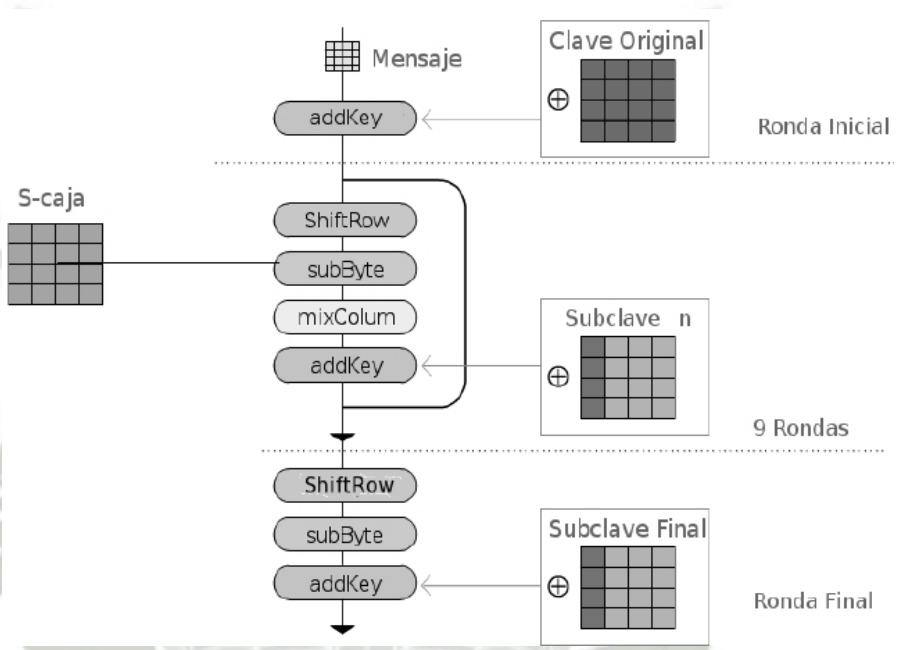

Figura 2.11: Diagrama del Funcionamiento del Algoritmo AES

(Lucena Lopez, 2010)

Ataques al Algoritmo AES

Ataque de temporizador de cache. Este tipo de ataque no afecta al algoritmo, sino más bien a una mala implementación de él, ya que va analizando la información encriptada y transmitida por medio de un servidor SSL.(Lucena Lopez, 2010)

2.3.1.4 Matrices de Hill

Los cifrados monográficos, en los que se sustituye un carácter por otro de una forma preestablecida, son vulnerables al análisis de frecuencia de aparición de las letras. Para evitarlo se desarrollaron esquemas basados en cifrar bloques de letras de una cierta longitud fija, o sea, cifrado poligráficos(Cipher A., Kahn, Kruh, & Mellen, 1987). El esquema que aquí trataremos se debe a Hill(Cipher A., Kahn, Kruh, & Mellen, 1987). Tiene un interés didáctico importante debido al uso de matrices que en él se hace.

Un cifrado de Hill se obtiene al transformar bloques de **n** caracteres en un texto cifrado a través de la relación

C = (A · P + B)(mod 28), donde:

- **A** es una matriz **n**x**n**, que debe ser inversible módulo 28, es decir, el m.c.d (determinante de_la_matriz A, 28)=1.
- **P** es un bloque de **n** caracteres. P = Z_{28}^{n}
- **B** es una matriz**nx1**
- **C** es la matriz columna resultante del cifrado de **P**. C = Z_{28}^{n}
- **28** es el número de símbolos del alfabeto: _ABCDEFGHIJKLMNÑOPQRSTUVWXY Z que se corresponden con los números del 0 al 27 (el 0 corresponde al espacio en blanco separador de dos palabras)

Un ejemplo para un cifrado digráfico (bloques de 2 caracteres) sería para el texto original siguiente: ESTACION CENTRAL X

E	S	T	A	C	I	O	N		C	E	N	T	R	A	L		X
5	20	21	1	3	9	16	14	0	3	5	14	21	19	1	12	0	25

Disponemos el texto de la forma siguiente y aplicamos la transformación indicada:

E	T	C	O		E	T	A	
S	A	I	N	C	N	R	L	X

Tomando como A la matriz $\begin{pmatrix} 1 & 27 \\ 0 & 3 \end{pmatrix}$; y como B la matriz $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ hacemos:

C1= (((1 * P1) + (27 * P2))+1) (mod 28)

C2 = (((0 * P1) + (3 * P2))+0) (mod 28)

Donde **P1** y **P2** son dos caracteres del mensaje sin cifrar, **C1** Y **C2** los correspondientes cifrados y **K**.

Continuando con el ejemplo y codificando

E
S

Siendo E = 5 y S = 20, entonces:

C1= (((1 * 5) + (27 * 20) + 1)(mod 28) = 546 (mod 28) = 14 (mod 28) (letra N)

C2= (((0 * 5) + (3 * 20) + 0))(mod 28) = 60 (mod 28) = 4 (mod 28) (letra D)

$$\begin{pmatrix} C_1 \\ C_2 \end{pmatrix} = \begin{pmatrix} 1 & 27 \\ 0 & 3 \end{pmatrix} \begin{pmatrix} P_1 \\ P_2 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \end{pmatrix} \text{(mod 28)}$$

Y así sucesivamente para cada bloque de 2 caracteres, resultando:

Texto cifrado: NDTCVZCNYISNCAQHDR

La consecuencia es que el mismo carácter se codifica de distintas formas (la primera E se ha codificado como una N, y la segunda E del texto original se ha codificado como una S).

El descifrado del sistema de Hill es simétrico (la clave de desencriptación se calcula a partir de la clave de encriptación y viceversa) y será aplicar la transformación:

$P = A^{-1} \cdot (C - B) \pmod{28}$, donde A^{-1} es la matriz inversa de A mod 28

Para calcular la inversa módulo **N** de una matriz cualquiera:Si **A** es una matriz tal que m.c.d(det(**A**), **N**)=1 y llamamos **d**=det(**A**) entonces $\mathbf{A^{-1}}=\mathbf{d^{-1}}.\mathbf{B}$ donde $\mathbf{d^{-1}}$ es el inverso de **d** módulo **N** y **B** es la transpuesta de la matriz adjunta de **A**.

Vamos a cifrar la palabra MAX utilizando un cifrado poligráfico de tamaño 3. Tomemos sus equivalentes numéricos:

M A X

13 1 25

Si las matrices de cifrado A y B son:

$$A = \begin{pmatrix} -1 & 0 & 3 \\ 1 & 1 & 2 \\ 1 & 1 & -1 \end{pmatrix} = \begin{pmatrix} 27 & 0 & 3 \\ 1 & 1 & 2 \\ 1 & 1 & 27 \end{pmatrix}; \quad B = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

Det(A) = d = 3 entonces tendremos que d^{-1}= 19 pues 3.19 = 57 = 1(mod 28); por otro lado

$$\text{Adj}(A) = = \begin{pmatrix} -3 & 3 & 0 \\ 3 & -2 & 1 \\ -3 & 5 & -1 \end{pmatrix} = \begin{pmatrix} 25 & 3 & 0 \\ 3 & 26 & 1 \\ 25 & 5 & 27 \end{pmatrix}; \quad \text{Adj } (A)^t = \begin{pmatrix} -3 & 3 & -3 \\ 3 & -2 & 5 \\ 0 & 1 & -1 \end{pmatrix}$$

Por lo que A^{-1}= 19. $\begin{pmatrix} 25 & 3 & 0 \\ 3 & 26 & 1 \\ 25 & 5 & 27 \end{pmatrix} = \begin{pmatrix} -1 & 1 & -1 \\ 1 & -10 & 11 \\ 0 & 19 & -19 \end{pmatrix} = \begin{pmatrix} 27 & 1 & 27 \\ 1 & 18 & 11 \\ 0 & 19 & 9 \end{pmatrix}$

Para cifrar $\begin{pmatrix} C_1 \\ C_2 \\ C_3 \end{pmatrix} = \begin{pmatrix} -1 & 0 & 3 \\ 1 & 1 & 2 \\ 1 & 1 & -1 \end{pmatrix} \cdot \begin{pmatrix} 13 \\ 1 \\ 25 \end{pmatrix} = \begin{pmatrix} 6 \\ 8 \\ 17 \end{pmatrix}$

El resultado es [62 64 -11] es decir [18 817] tomándolo módulo 26. Así que el bloque que corresponde a MAX es QHP.

Probemos con el descifrado:

$$\begin{pmatrix} P_1 \\ P_2 \\ P_3 \end{pmatrix} = \begin{pmatrix} -1 & 1 & -1 \\ 1 & -10 & 11 \\ 0 & 19 & -19 \end{pmatrix} \begin{pmatrix} 6 \\ 8 \\ 17 \end{pmatrix} = \begin{pmatrix} 13 \\ 1 \\ -3 \end{pmatrix}$$

El resultado es como era de prever [13 1 25], es decir el bloque MAX original.(Cipher A., Kahn, Kruh, & Mellen, 1987)

Ataques al Algoritmo de Hill

El método consiste en escribir una matriz 2N-grámica con los elementos del texto en claro y los elementos del criptograma. En esta matriz realizamos operaciones lineales (multiplicar filas por un número y restar filas entre sí) con el objeto de obtener los vectores unitarios. Por ejemplo podemos romper la matriz clave K teniendo:

M = ENU NLU GAR DEL AMA NCH ADE CUY ONO ...

C =WVX IDQ DDO ITQ JGO GJI YMG FVC UÑT ...

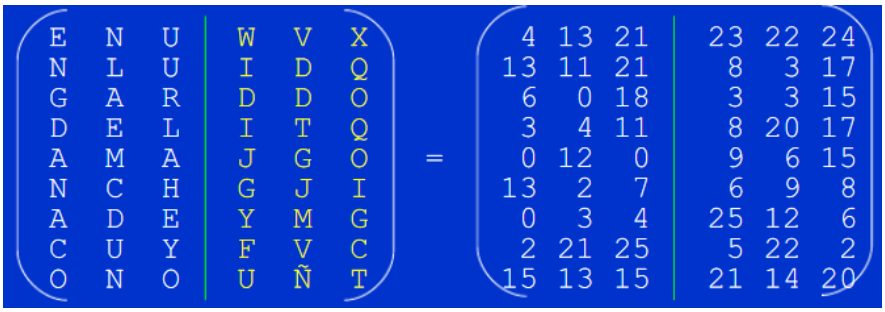

Figura 2.12: Ataque al Cifrado de Hill por Gauss Jordán

(Gordon, 1993)

Vamos a dejar en la primera columna un número uno en la fila primera y todas las demás filas un cero. Luego multiplicamos el vector (413 21 | 23 22 24) por el inv (4, 27) = 7. Así obtenemos 7(4 13 21 | 23 22 24) mod 27 = (1 10 12 | 26 19 6). Si esto no se puede hacer con la primera fila movemos los vectores. Hecho estovamos restando las filas respecto de esta primera como se indica:

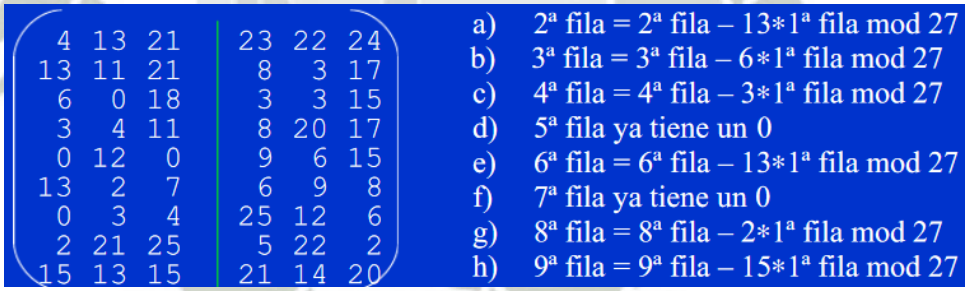

Figura 2.13: Representación de cada Fila

(Gordon, 1993)

Repetimos este procedimiento ahora para algún vector en cuya segunda columna tenga un número con inverso en 27 y lo mismo para la tercera columna, moviendo si es preciso los vectores. Como la mitad izquierda de la matriz 2N era el texto el claro, la

parte derecha de la matriz con vectores unitarios corresponderá a la traspuesta de la clave.

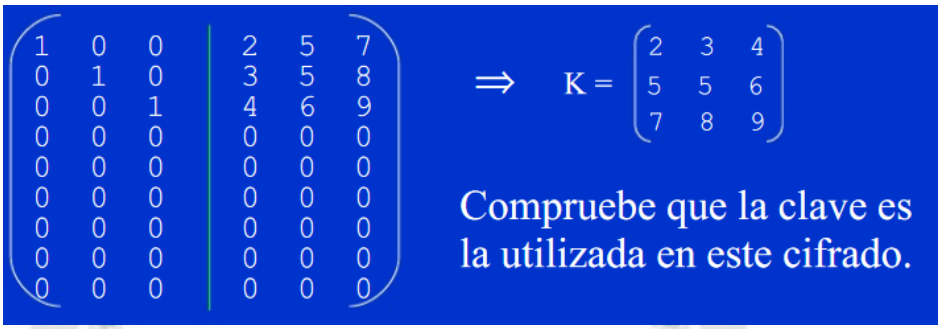

Figura 2.14: Clave que se Utilizara para el Cifrado

(Gordon, 1993)

2.4 ALGORITMOS GENETICOS

El algoritmo genético es una técnica de búsqueda basada en la teoría de la evolución de Darwin, que ha cobrado tremenda popularidad en todo el mundo durante los últimos años. Se presentarán aquí los conceptos básicos que se requieren para abordarla, así como unos sencillos ejemplos que permitan a los lectores comprender cómo aplicarla al problema de su elección.(Mitchell, 1999)

Es un algoritmo matemático altamente paralelo que transforma un conjunto de objetos matemáticos individuales con respecto al tiempo usando operaciones modeladas de acuerdo al principio Darwiniano de reproducción y supervivencia del más apto, y tras haberse presentado de forma natural una serie de operaciones genéticas de entre las que destaca la recombinación sexual. Cada uno de estos objetos matemáticos suele ser una cadena de caracteres (letras o números) de longitud fija que se ajusta al modelo de las cadenas de cromosomas, y se les asocia con una cierta función matemática que refleja su aptitud.(Haupt & Haupt, 2004)

Los Algoritmos Genéticos (AGs) son métodos adaptativos que pueden usarse para resolver problemas de búsqueda y optimización. Están basados en el proceso genético de los organismos vivos. A lo largo de las generaciones, las poblaciones evolucionan en la naturaleza de acorde con los principios de la selección natural y la supervivencia de los más fuertes, postulados por Darwin. Por imitación de este proceso, los Algoritmos Genéticos son capaces de ir creando soluciones para problemas del mundo real. La evolución de dichas soluciones hacia valores óptimos del problema depende en buena medida de una adecuada codificación de las mismas.
Un algoritmo genético consiste en una función matemática o una rutina de software que toma como entradas a los ejemplares y retorna como salidas cuáles de ellos deben generar descendencia para la nueva generación.(Mitchell, 1999)

Versiones más complejas de algoritmos genéticos generan un ciclo iterativo que directamente toma a la especie (el total de los ejemplares) y crea una nueva generación que reemplaza a la antigua una cantidad de veces determinada por su propio diseño. Una de sus características principales es la de ir perfeccionando su propia heurística en el proceso de ejecución, por lo que no requiere largos períodos de entrenamiento especializado por parte del ser humano, principal defecto de otros métodos para solucionar problemas, como los Sistemas Expertos.

2.5 INTELIGENCIA ARTIFICIAL

La Inteligencia Artificial es una combinación de la ciencia del computador, fisiología y filosofía, tan general y amplio como eso, es que reúne varios campos (robótica, sistemas expertos, por ejemplo), todos los cuales tienen en común la creación de máquinas que pueden pensar. (Haupt & Haupt, 2004)

La idea construir una máquina que pueda ejecutar tareas percibidas como requerimientos de inteligencia humana es un atractivo. Las tareas que han sido estudiadas desde este punto de vista incluyen juegos, traducción de idiomas, comprensión de idiomas, diagnóstico de fallas, robótica, suministro de asesoría experta en diversos temas.

Es así como los sistemas de administración de base de datos cada vez más sofisticados, la estructura de datos y el desarrollo de algoritmos de inserción, borrado y locación de datos, así como el intento de crear máquinas capaces de realizar tareas que son pensadas como típicas del ámbito de la inteligencia humana, acuñaron el término Inteligencia Artificial en 1956.

La Inteligencia Artificial trata de conseguir que los ordenadores simulen en cierta manera la inteligencia humana(Haupt & Haupt, 2004). Se acude a sus técnicas cuando es necesario incorporar en un sistema informático, conocimiento o características propias del ser humano.
Podemos interrogar a algunas bases de datos de Internet en lenguaje natural, o incluso charlar con ellas nuestro idioma, porque por detrás se está ejecutando un programa de Inteligencia Artificial.
Otras herramientas inteligentes pueden utilizarse para escrutar entre los millones de datos que se generan en un banco en busca de patrones de comportamiento de sus clientes o para detectar tendencias en los mercados de valores.

CAPITULO III:

DESARROLLO DE LA PROPUESTA

En esta etapa se inicia la elaboración del software. La característica principal del desarrollo de software es que sigue una metodología incremental, es decir que el proyecto se divide en diferentes etapas, llevando a cabo una por una y dejando las demás para niveles posteriores, hasta cumplir con todos los requerimientos establecidos.

3.1 DEFINICION DEL SOFTWARE DE DATOS UTILIZANDO TÉCNICAS GENETICAS

El software de encriptación de datos implementa cualquier protocolo criptográfico incrementando su eficiencia al utilizar técnicas genéticas para encriptar y desencriptar información significativa para el usuario, esto quiere decir que el software permite editar mensajes.

La interfaz del software contiene un editor, donde se lleva a cabo todas las alternativas mencionadas, los dispositivos de entrada son los mensajes originales que el usuario desea enviar a un receptor o simplemente desea

guardarlo de forma segura, para lo cual ocurre el proceso de encriptamiento mediante técnicas genéticas

3.2 SELECCIÓN DE PLATAFORMA Y HERRAMIENTAS DE DESARROLLO

La selección del sistema operativo (o plataforma) sobre la que operaría el software de encriptación de datos, se hizo teniendo en cuenta los siguientes criterios:

- Difusión: (la plataforma seleccionada debe ser ampliamente conocida y utilizada por los usuarios).
- Potencia y flexibilidad (facilidades propias de la plataforma, para el desarrollo del software diseñado).

Será implementadas bajo los sistemas operativos de Windows XP y 7 ya que cumple con todos los criterios mencionados, son Sistemas Operativos de 32 bits o 64 bits, Multitarea real: control asíncrono, múltiples hebras de ejecución, Velocidad alta, Memoria plana direccionable directamente, API mejorada, que incluye multimedia, El S.O. gestiona a todos los dispositivos de E/S, Variedad de herramientas de desarrollo.

Ya definida la plataforma de trabajo, se procedió a seleccionar las herramientas de software más apropiadas para la elaboración del software de encriptamiento.

Para el desarrollo del código y generación del programa ejecutable, se evaluaron herramientas y un lenguaje de propósito general Visual C#; todas ellas para plataforma Windows 7

La decisión final fue utilizar la herramienta Microsoft Visual C# para el desarrollo del software, por los siguientes motivos:

- Es la más flexible de todas las evaluadas.

- Genera código ejecutable altamente optimado y veloz.
- Permite la programación orientada a objetos.

3.3 SECUENCIA DE CREACION DEL SOFTWARE

La secuencia de creación del software de encriptación de datos se basó en los lineamientos de prototipos utilizando la metodología incremental.

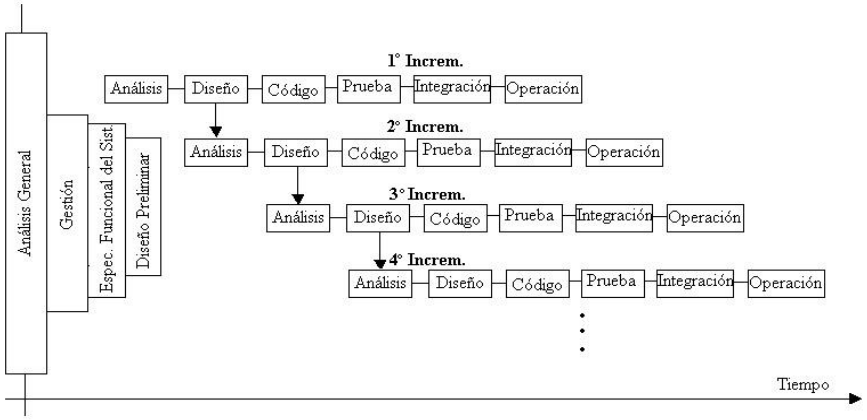

Figura 3.1: Metodología Incremental

(Fuente Propia)

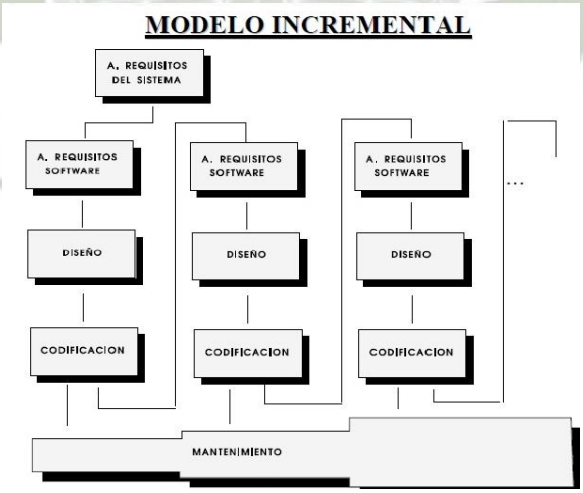

Figura 3.2: Metodología Incremental

(Fuente Propia)

Análisis General

Se desarrollara el software por aproximaciones sucesivas, comenzando por lo más importante. Las decisiones complementarias se van dejando para después, cuando se demuestre que hemos alcanzado lo esencial. Así reducimos nuestro riesgo de hacer algo más allá y tenerlo que desechar. Este será nuestro modo cíclico de actuar frente a la incertidumbre: reducir progresivamente el riesgo. Hacer apuestas pequeñas, contrario a la Cascada, que lo apuesta todo a un disparo, supuestamente certero. Para que el desarrollo sea evolutivo, el diseño tiene que serlo también.

Los principales lineamientos a seguir son:

3.3.1 ANÁLISIS

En esta primera fase realizaremos las operaciones de mayor interés siguiendo la línea de desarrollo evolutivo, es decir, crecimiento y mejora progresiva del software, partiremos de una situación simple que es el caso de la encriptación y desencriptación de la información. El software de encriptado y desencriptado trabajara con cualquier protocolo criptográfico como si fuera un juego de llaves que al enviar un nuevo mensaje utilice un protocolo criptográfico distinto al anterior, a su vez la elección de que protocolo criptográfico se utilice se realizara de manera aleatoria.

Para la realización del código fuente del prototipo se trabajara con protocolo de matrices de Hill, se optó por este algoritmo ya que su desarrollo es muy simple y nos proporciona la opción de no modificarlo al implementar algoritmos genéticos.

3.3.2 DISEÑO

Como el diseño es un acto creativo se nos pueden ocurrir muchas soluciones válidas; unas serán mejores que otras, de acuerdo a las condiciones específicas del problema y su entorno. Se sugiere conseguir diseños simples y sencillos. Para procurar hacerlo todo lo menos complicado posible para el usuario o cliente, para conseguir un diseño fácilmente entendible e impleméntable que a la larga costará menos tiempo y esfuerzo para desarrollarlo, como se quiere que el sistema sea lo más globalizado posible en cuanto a la elección del protocolo criptográfico a usarse todos los diagramas se realizaran de manera global sin tener mucho interés en la especificación o detalle de los protocolos criptográficos.

El software debe permitir que el emisor envié un mensaje de manera confidencial para que si el mensaje es interceptado por personas no autorizadas o desconocidas estos no puedan saber entender el contenido del mensaje. La Figura muestra una implementación en objetos del software descrito.

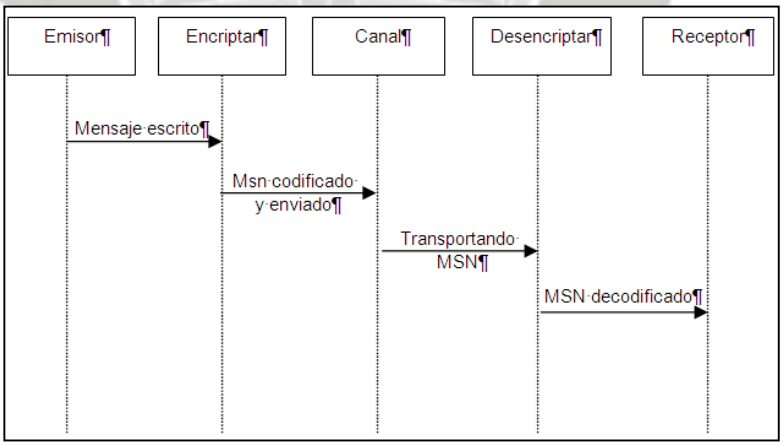

Figura 3.3: Diagrama de Secuencia de Envió y Recepción de Mensajes Encriptados

(Fuente Propia)

En esta implementación hemos diseñado el método MSN codificado el cual se encargara de codificar el mensaje escrito por el emisor utilizando un protocolo criptografía (matrices) para que luego se envié el mensaje codificado, una vez trasportado el mensaje este será recibido por el método MSN decodificado el cual realizara el proceso de decodificar el mensaje usando el mismo protocolo criptográfico utilizado en la codificación (matrices) para que el mensaje sea entendible para el receptor.

Siguiendo nuestro estilo de primero lo esencial indicaremos los objetos que se han diseñado para que participen en el software.

- Emisor: Encargo de enviar el mensaje.
- Encriptado: Su función es la de codificar el mensaje escrito por el emisor antes de ser enviado de tal manera que este sea ininteligible para intrusos (lectores no autorizados) que intercepten el mensaje, para poder tener confidencialidad en el mensaje se aplicara cualquier protocolo criptográfico que nos permita cumplir con el objetivo como ejemplo se utilizara el protocolo de matrices.
- Canal: Es el medio por el cual el emisor se comunicara con el receptor para poder enviar el mensaje por ejemplo internet, red local, etc.
- Desencriptado: Antes de que el mensaje sea leído por el receptor se recepciona y se procede a decodificarlo para que pueda ser entendible por el receptor, para la decodificación del mensaje se deberá utilizar el mismo protocolo criptográfico que se ha utilizado para la codificación., en este caso será el de matrices.
- Receptor: Recibe el mensaje

Resumiendo el encriptado es la pieza del software en cargada de codificar el mensaje escrito por el emisor antes de ser enviado por un canal de comunicación, el mensaje codificado brinda confidencialidad ya que dicho contenido será ininteligible para intrusos que intercepten el mensaje, el objeto desencriptar recepciona el mensaje y procede a decodificarlo para que pueda ser entendible por el receptor.

Para codificar y decodificar los mensajes se utilizan diferentes protocolos criptográficos los cuales brinda seguridad a la información que se envía, para este caso se utilizar el protocolo de matrices.

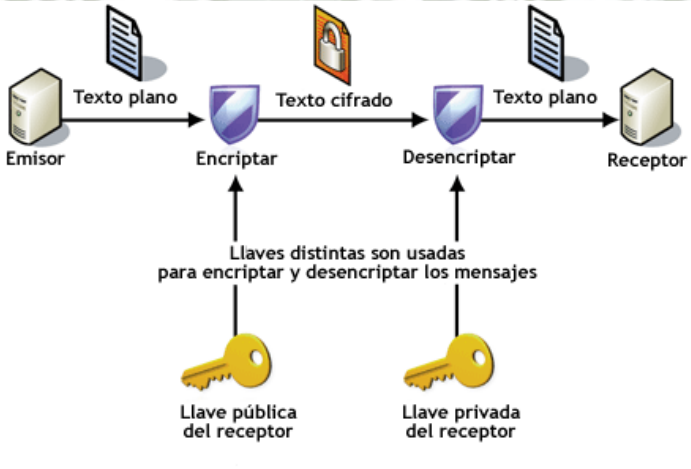

Figura 3.4: Esquema de Envió de Mensaje Encriptado

(Lucena Lopez, 2010)

3.3.2.1 Diagrama de clases del sistema de encriptado y desencriptado de mensajes

En el diagrama de clases del sistema de encriptado y desencriptado de mensajes se ilustran las clases de cada uno de

los objetos que participan en la construcción del mecanismo y las relaciones que existen entre ellas. Figura 5. Desde el punto de vista sintáctico, debe existir una línea entre clases si existe al menos un mensaje entre los objetos de esas clases.

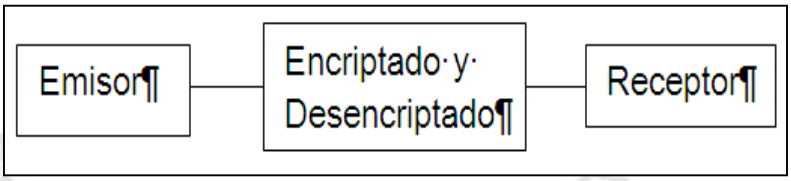

Figura 3.5: Diagrama de Clases del Sistema de Encriptación y Desencriptado de Mensajes

(Fuente Propia)

- La clase encriptado y desencriptado codifica el mensaje escrito por el emisor para luego ser enviado al receptor, este último al recibir el mensaje la clase decodifica el mensaje para luego ser leído por el receptor.
- El emisor como el receptor pueden ser interpretados como usuarios de un sistema, cliente-servidor, etc.

3.3.2.2 Diagrama de caso de uso del sistema de encriptado y desencriptado de mensajes

El diagrama de casos de usos es una secuencia de interacciones que se desarrollarán entre un sistema y sus actores en respuesta a un evento que inicia un actor principal sobre el propio sistema. Los diagramas de casos de uso sirven para especificar la comunicación y el comportamiento de un sistema mediante su

interacción con los usuarios y/u otros sistemas. O lo que es igual, un diagrama que muestra la relación entre los actores y los casos de uso en un sistema. Una relación es una conexión entre los elementos del modelo, por ejemplo la especialización y la generalización son relaciones. Los diagramas de casos de uso se utilizan para ilustrar los requerimientos del sistema al mostrar cómo reacciona a eventos que se producen en su ámbito o en él mismo

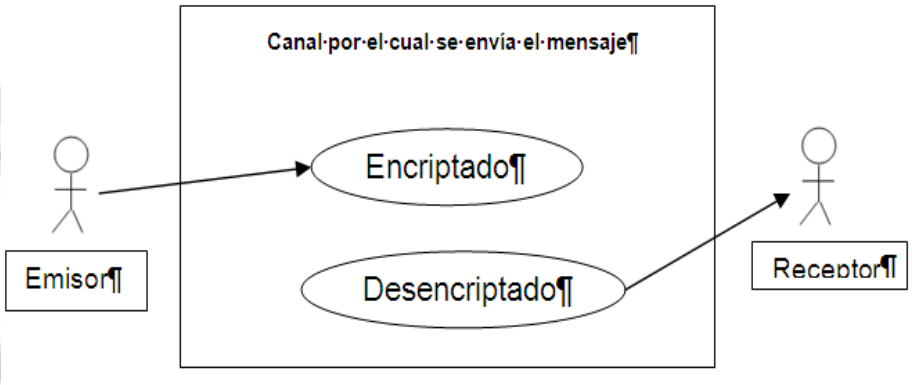

Figura 3.6: Diagrama de Caso de Uso del Sistema de Encriptado y Desencriptado de Mensajes

(Fuente Propia)

Tabla 3.1 Tabla de Caso de Uso de la Función Encriptado

Código	Cs-001
Nombre	Encriptado
Actores	Emisor
Descripción	El mensaje escrito por el emisor será codificado para luego ser enviado.
Precondición	El emisor debe haber escrito el mensaje
PostCondiciones	Ninguna

Tabla 3.2 Tabla de Caso de Uso de la Función Desencriptado

Código	Cs-002
Nombre	Desencriptado
Actores	Receptor
Descripción	Recibe el mensaje para su decodificación para que el receptor pueda leerlo.
Precondición	El mensaje debe estar codificado
PostCondiciones	Ninguna

Flujo Principal

1. Emisor: Redacta un mensaje
2. Sistema: El mensaje es codificado y enviado por un canal de comunicación
3. Sistema: El mensaje es recibido y decodificado.
4. Receptor: El mensaje es leído

3.3.2.3 Diagrama de Estado

El diagrama de estado nos permite ver el comportamiento del sistema. Describiendo todos los estados posibles en los que puede entrar un objeto particular y la manera en que cambia el estado del objeto, como resultado de los eventos que llegan a él.

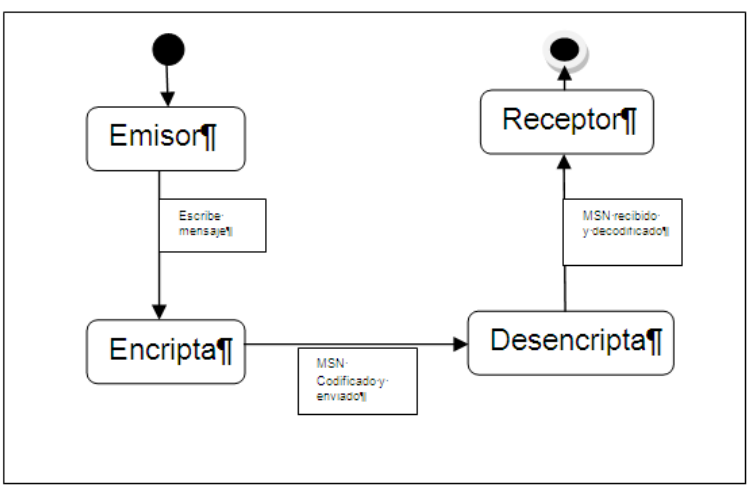

Figura 3.7: Diagrama de Estado del Sistema de Encriptado y Desencriptado de Mensajes

(Fuente Propia)

Por último en esta fase se logrará crear parte del proyecto la parte física (lo bonito) la interfaz que tendrá el usuario o cliente con el proyecto.

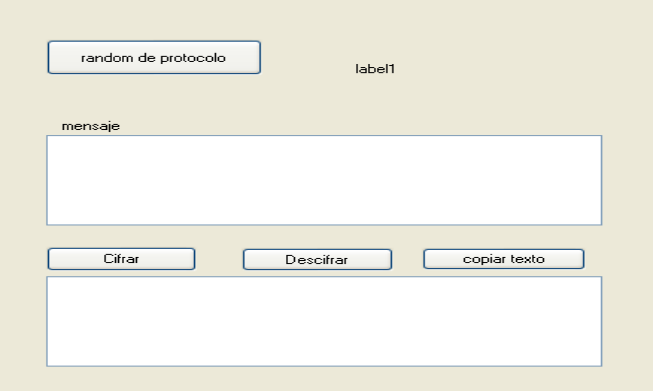

Figura 3.8: Entorno Grafico del Prototipo

(Fuente Propia)

3.3.3 CODIFICACIÓN

Hecho el diseño procedemos a la codificación del software para su primera presentación al cliente.

Como indicamos en el análisis el sistema está dado para poder trabajar con diferentes protocolos criptográficos pero para su codificación y prueba solo se utilizara el protocolo criptográfico matrices.

3.3.3.1 IDEA DEL ALGORITMO

Como se indicó en el capítulo 2 el desarrollo del algoritmo de cifrado es simple. Un cifrado de Hill se obtiene al transformar bloques de **n** caracteres en un texto cifrado a través de la relación

C = (A · P + B)(mod 28), donde:

- **A** es una matriz **n**x**n**, que debe ser inversible módulo 28, es decir, el m.c.d (determinante de_la_matriz [][]A, 28)=1.
- **P** es un bloque de **n** caracteres. P = Z_{28}^{n}
- **B** es una matriz**nx1**
- **C** es la matriz columna resultante del cifrado de **P**. C = Z_{28}^{n}
- **28** es el número de símbolos del alfabeto: _ABCDEFGHIJKLMNÑOPQRSTUVWXY Z que se corresponden con los números del 0 al 27 (el 0 corresponde al espacio en blanco separador de dos palabras)

Un ejemplo para un cifrado digráfico (bloques de 2 caracteres) sería para el texto original siguiente: ESTACION CENTRAL X

E	S	T	A	C	I	O	N		C	E	N	T	R	A	L		X
5	20	21	1	3	9	16	14	0	3	5	14	21	19	1	12	0	25

Disponemos el texto de la forma siguiente y aplicamos la transformación indicada:

E	T	C	O		E	T	A	
S	A	I	N	C	N	R	L	X

Tomando como A la matriz $\begin{pmatrix} 1 & 27 \\ 0 & 3 \end{pmatrix}$; y como B la matriz $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ hacemos:

C1= (((1 * P1) + (27 * P2))+1) (mod 28)

C2 = (((0 * P1) + (3 * P2))+0) (mod 28)

Donde **P1** y **P2** son dos caracteres del mensaje sin cifrar, **C1** Y **C2** los correspondientes cifrados y **K**.

Continuando con el ejemplo y codificando

Siendo E = 5 y S = 20, entonces:

C1= (((1 * 5) + (27 * 20) + 1)(mod 28) = 546 (mod 28) = 14 (mod 28) (letra N)

C2= (((0 * 5) + (3 * 20) + 0))(mod 28) = 60 (mod 28) = 4 (mod 28) (letra D)

$$\begin{pmatrix} C_1 \\ C_2 \end{pmatrix} = \begin{pmatrix} 1 & 27 \\ 0 & 3 \end{pmatrix} \begin{pmatrix} P_1 \\ P_2 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \end{pmatrix} (\text{mod } 28)$$

Y así sucesivamente para cada bloque de 2 caracteres, resultando:

Texto cifrado: NDTCVZCNYISNCAQHDR

La consecuencia es que el mismo carácter se codifica de distintas formas (la primera E se ha codificado como una N, y la segunda E del texto original se ha codificado como una S).

El descifrado del sistema de Hill es simétrico (la clave de desencriptación se calcula a partir de la clave de encriptación y viceversa) y será aplicar la transformación:

$P = A^{-1} \cdot (C - B) \pmod{28}$, donde A^{-1} es la matriz inversa de A mod 28

Para calcular la inversa módulo **N** de una matriz cualquiera:Si **A** es una matriz tal que m.c.d(det(**A**), **N**)=1 y llamamos **d**=det(**A**) entonces $\mathbf{A^{-1}}=\mathbf{d^{-1}}.\mathbf{B}$ donde $\mathbf{d^{-1}}$ es el inverso de **d** módulo **N** y **B** es la transpuesta de la matriz adjunta de **A**.

Vamos a cifrar la palabra MAX utilizando un cifrado poligráfico de tamaño 3. Tomemos sus equivalentes numéricos:

M A X

13 1 25

Si las matrices de cifrado A y B son:

$$A = \begin{pmatrix} -1 & 0 & 3 \\ 1 & 1 & 2 \\ 1 & 1 & -1 \end{pmatrix} = \begin{pmatrix} 27 & 0 & 3 \\ 1 & 1 & 2 \\ 1 & 1 & 27 \end{pmatrix}; B = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

Det(A) = d = 3 entonces tendremos que d^{-1}= 19 pues 3.19 = 57 = 1(mod 28); por otro lado

$$Adj(A) = = \begin{pmatrix} -3 & 3 & 0 \\ 3 & -2 & 1 \\ -3 & 5 & -1 \end{pmatrix} = \begin{pmatrix} 25 & 3 & 0 \\ 3 & 26 & 1 \\ 25 & 5 & 27 \end{pmatrix}; Adj\ (A)^t = \begin{pmatrix} -3 & 3 & -3 \\ 3 & -2 & 5 \\ 0 & 1 & -1 \end{pmatrix}$$

Por lo que $A^{-1}= 19.\begin{pmatrix} 25 & 3 & 0 \\ 3 & 26 & 1 \\ 25 & 5 & 27 \end{pmatrix} = \begin{pmatrix} -1 & 1 & -1 \\ 1 & -10 & 11 \\ 0 & 19 & -19 \end{pmatrix} = \begin{pmatrix} 27 & 1 & 27 \\ 1 & 18 & 11 \\ 0 & 19 & 9 \end{pmatrix}$

Para cifrar $\begin{pmatrix} C_1 \\ C_2 \\ C_3 \end{pmatrix} = \begin{pmatrix} -1 & 0 & 3 \\ 1 & 1 & 2 \\ 1 & 1 & -1 \end{pmatrix} \cdot \begin{pmatrix} 13 \\ 1 \\ 25 \end{pmatrix} = \begin{pmatrix} 6 \\ 8 \\ 17 \end{pmatrix}$

El resultado es [62 64 -11] es decir [18 817] tomándolo módulo 26. Así que el bloque que corresponde a MAX es QHP.

Probemos con el descifrado:

$$\begin{pmatrix} P_1 \\ P_2 \\ P_3 \end{pmatrix} = \begin{pmatrix} -1 & 1 & -1 \\ 1 & -10 & 11 \\ 0 & 19 & -19 \end{pmatrix} \begin{pmatrix} 6 \\ 8 \\ 17 \end{pmatrix} = \begin{pmatrix} 13 \\ 1 \\ -3 \end{pmatrix}$$

El resultado es como era de prever [13 1 25], es decir el bloque MAX original.(Cipher A., Kahn, Kruh, & Mellen, 1987)

3.3.3.2 Código Fuente

```
classmatriz
   {

char[] Letras = { 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'Ñ', 'O',
'P','Q','R','S','T','U','V','W','X','Y','Z',' ','.',',','¿','?',' ' };
char[] LetrasM = { 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'ñ', 'o', 'p', 'q', 'r', 's', 't',
'u', 'v', 'w', 'x', 'y', 'z' ,'.',',','¿','?',' '};
int[] Numeros;
int [,] clave = {{2,17},{3,5}};
int[,] ClaveInv = { { 3, 9 }, { 11, 14 } };

publicmatriz()
      {

int[] Numeros = newint[Letras.Length];
for (inti = 0; i<Letras.Length; i++)
{
Numeros[i] = i;

         }
```

```
this.Numeros=Numeros;
      }
```

Código de cifrado de mensaje

```
publicstringCifrar(stringmensaje)
{

char[] mensajecifrado = mensaje.ToCharArray();
string MSN ;
int [] msncifrado = newint[mensaje.Length];
int[] pcifrado;
int j;
int k;
for (inti = 0; i<mensaje.Length; i++)
          {
              j = 0;
              k = 0;
do
              {
if (mensajecifrado[i] == Letras[k] || mensajecifrado[i] == LetrasM[k])
                  {
                      j = k;
break;
                  }
k++;
          } while (mensajecifrado[i] != Letras[k]);

msncifrado[i] = Numeros[k];
}



if (msncifrado.Length % 2 == 0)
          {
pcifrado = newint[mensaje.Length];
for (inti = 0; i<msncifrado.Length; i++)
{
pcifrado[i] = (clave[0, 0] * msncifrado[i]) + (clave[0, 1] * msncifrado[i + 1]);
pcifrado[i + 1] = (clave[1, 0] * msncifrado[i]) + (clave[1, 1] * msncifrado[i + 1]);
i++;
              }


          }

else

          {
pcifrado = newint[msncifrado.Length+1];
int[] msncifradoNew = newint[msncifrado.Length+1];
for (inti = 0; i<msncifrado.Length + 1; i++)
              {

if (i == msncifrado.Length)
                  {
msncifradoNew[i] = 31;
                  }
```

```
else { msncifradoNew[i] = msncifrado[i]; }
            }

for (inti = 0; i<msncifrado.Length; i++)
            {
pcifrado[i] = (clave[0, 0] * msncifradoNew[i]) + (clave[0, 1] * msncifradoNew[i + 1]);
pcifrado[i + 1] = (clave[1, 0] * msncifradoNew[i]) + (clave[1, 1] * msncifradoNew[i + 1]);
i++;
            }

        }

char[] cifrado = newchar[pcifrado.Length];

for (inti = 0; i<pcifrado.Length; i++)
{
int t = (pcifrado[i])%32;
cifrado[i] = Letras[t];
}

MSN = newstring(cifrado);
return (MSN);

}
```

Código de descifrado de mensaje

```
publicstring Descifrar(string mensaje)
      {
char[] mensajecifrado = mensaje.ToCharArray();
string MSN;
int[] msncifrado = newint[mensaje.Length];
int[] pcifrado;
int j;
int k;
for (inti = 0; i<mensaje.Length; i++)
        {
            j = 0;
            k = 0;
do
            {
if (mensajecifrado[i] == Letras[k])
                {
                    j = k;
break;
                }
if (mensajecifrado[i] == LetrasM[k])
                {
                    j = k;
break;
                }
k++;
            } while (mensajecifrado[i] != Letras[k]);

msncifrado[i] = Numeros[k];
}
```

```
pcifrado = newint[mensaje.Length];
for (inti = 0; i<msncifrado.Length; i++)
{
pcifrado[i] = Math.Abs((ClaveInv[0, 0] * msncifrado[i]) + (ClaveInv[0, 1] * msncifrado[i +
1]));
pcifrado[i + 1] = Math.Abs((ClaveInv[1, 0] * msncifrado[i]) + (ClaveInv[1, 1] *
msncifrado[i + 1]));
i++;
            }


char[] Descifrado = newchar[pcifrado.Length];

for (inti = 0; i<pcifrado.Length; i++)
{
int t = (pcifrado[i]) % 32;
Descifrado[i] = Letras[t];
}


        MSN = newstring(Descifrado);

return (MSN);

}
```

3.3.4 PRIMERA AMPLIACIÓN: IMPLEMENTAR TÉCNICAS GENÉTICAS

En el prototipo anterior nos enfocamos en codificar y decodificar un mensaje utilizando protocolos criptográficos (cifrado de Hill), ahora en esta parte incrementaremos la seguridad de la codificación implementando técnicas genéticas las cuales nos brindaran una capa de seguridad adicional a la información enviada, así mismo utilizaremos las técnicas genéticas para el proceso de decodificación.

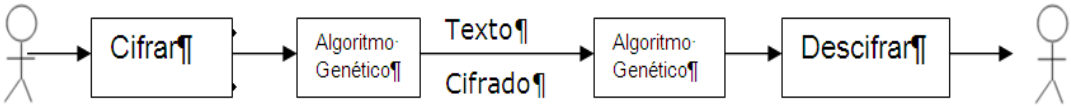

Figura 3.9: Sistema de Encriptado Utilizando Técnicas Genéticas

(Fuente Propia)

3.3.4.1 DISEÑO

Se implementara un nuevo objeto al diseño anterior el cual son las técnicas genéticas los cuales nos permitirán incrementar la seguridad de la información enviada, así siguiendo la mecánica del diseño anterior el emisor escribiría un mensaje el cual se codificaría con el protocolo criptográfico elegido y una vez codificado el mensaje a este se le aplicaría técnicas genéticas las cuales son selección, combinación y mutación que resultarían de dar un nuevo mensaje codificado una vez terminado el nuevo mensaje se procederá a su envió por medio de cualquier canal de comunicación, al llegar a su destino este comenzara primero aplicando técnicas genéticas para volver a recuperar el mensaje codificado para luego ser decodificado utilizando el protocolo criptográfico elegido antes de su envió y por ultimo ser leído por el receptor.

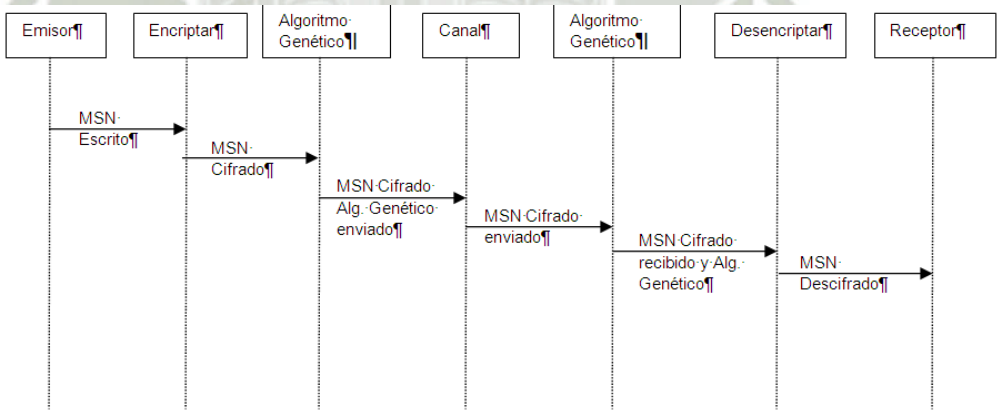

Figura 3.10: Diagrama de Secuencia de Sistema de Encriptado Utilizando Técnicas Genéticas

(Fuente Propia)

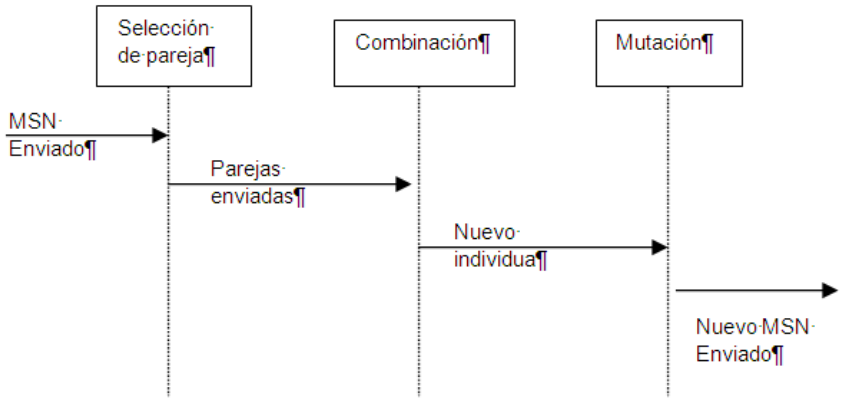

Figura 3.11: Diagrama de Secuencia de las Técnicas Genéticas

(Fuente Propia)

Como vemos en la figura el mensaje codificado pasa por una selección de parejas, después se realiza una combinación con las parejas escogidas para obtener un nuevo individuo por último se realiza la mutación la cual consiste en cambiar un valor aleatorio del nuevo individuo con lo cual obtenemos un nuevo mensaje el cual es enviado por el canal de comunicación escogido, para la decodificación es el mismo proceso.

3.3.4.1.1 Diagrama de clase

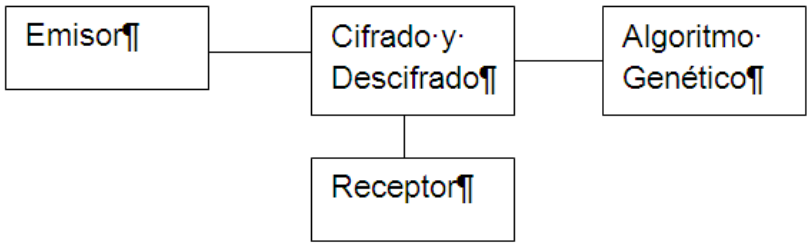

Figura 3.12: Diagrama de Clases de Sistema de Cifrado con Técnicas Genéticas

(Fuente Propia)

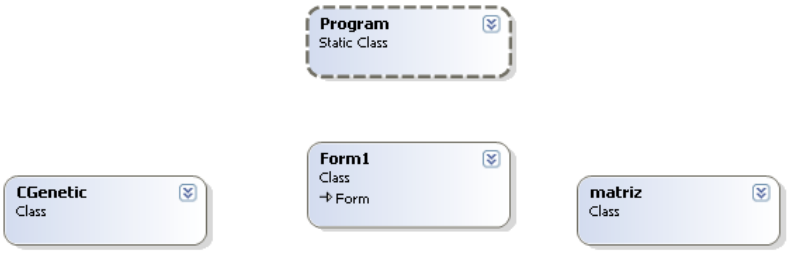

Figura 3.13: Diagrama de Clases del Software

(Fuente Propia)

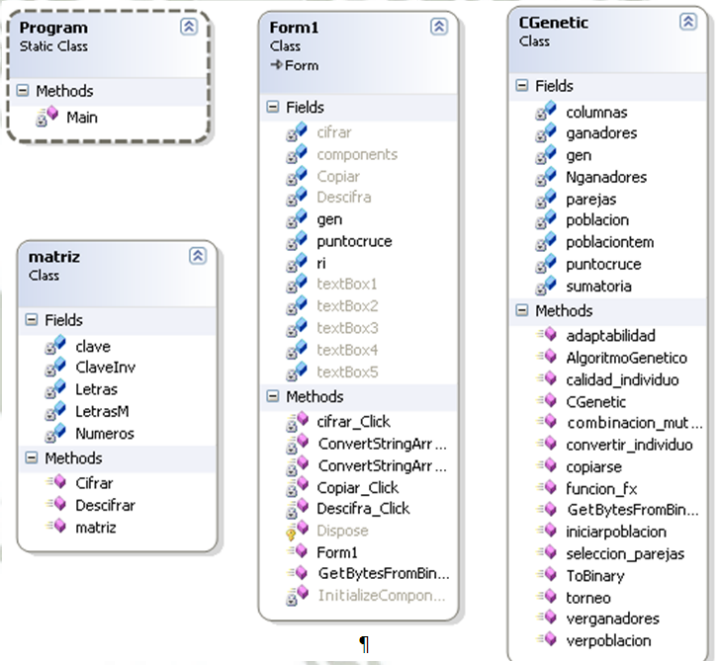

Figura 3.14: Diagrama de Clases Desglosado

(Fuente Propia)

3.3.4.1.2 Diagrama de Caso de uso

El siguiente diagrama se modificó al implementar las técnicas genéticas dando como resultado un diagrama de caso de uso.

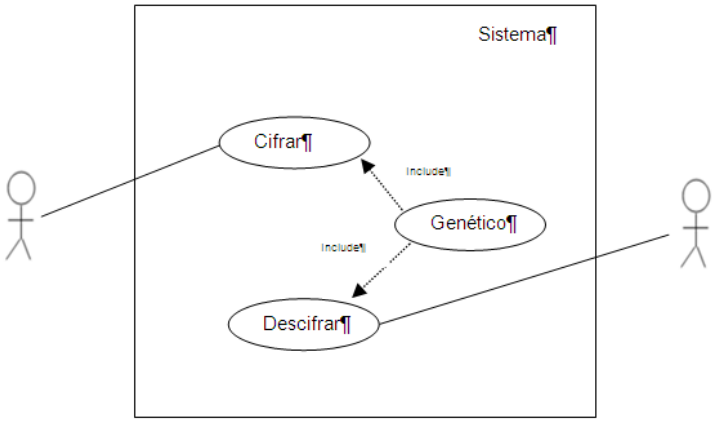

Figura 3.15: Diagrama de Caso de Uso

(Fuente Propia)

Caso de Uso Extendido:

Tabla 3.3 Tabla de Caso de Uso de la Función Cifrar

Caso de uso:	Cifrar
Actores:	Emisor
Propósito:	Codificar el mensaje antes de enviarlo
Resumen:	El emisor redacta el mensaje para luego poder enviarlo
Tipo:	Primario
Pre condiciones:	Se debe de haber escrito algún mensaje
Curso Normal de Eventos	
Acción del Autor	**Respuestas del Sistema**
1. Este caso de uso	**2.**Muestra el mensaje codificado

empieza cuando el emisor esta por enviar un mensaje, el mensaje es codificado utilizando cualquier protocolo criptográfico. 3. Se envía al algoritmo genético	
Sección Genetico	
Acción del Autor	**Respuestas del Sistema**
	1. Se recepciona el mensaje codificado para luego modificarse 2. Se aplica algoritmo genético 3. se envía el nuevo mensaje
Flujo Alterno	

Tabla 3.4 Tabla de Caso de Uso de la Función Descifra

Caso de uso:	Descifrar
Actores:	Receptor
Propósito:	Decodificar el mensaje antes de enviarlo
Resumen:	Se recepciona el mensaje para ser decodificado
Tipo:	Primario
Pre condiciones:	Debe haber un mensaje cifrado
Curso Normal de Eventos	
Acción del Autor	**Respuestas del Sistema**
1. Este caso de uso empieza cuando el mensaje codificado es recibido. 3. Se envía al algoritmo genético 4. se recepciona para luego proceder a su decodificación aplicando el protocolo criptográfico elegido en un principio	**2.** Muestra el mensaje codificado

Sección Genetico	
Acción del Autor	**Respuestas del Sistema**
	1.Se recepciona el mensaje codificado para luego modificarse 2. Se aplica algoritmo genético 3. se envía el nuevo mensaje
Flujo Alterno	

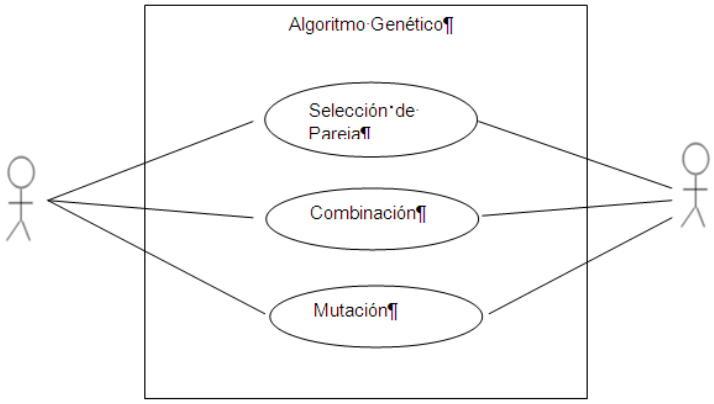

Figura 3.16: Diagrama de Caso de Uso del Algoritmo Genético

(Fuente Propia)

Tabla 3.5 Tabla de Caso de Uso de la Función Selección de Pareja

Caso de uso:	Seleccion de pareja
Actores:	Mensaje enviado/recibido
Propósito:	Seleccionar una pareja para proceder a la combinación
Resumen:	El mensaje codificado se convierte en binario para luego ser dividido en pares de 8 bits y luego ser emparejado
Tipo:	Primario
Pre condiciones:	mensaje cifrado convertido en binario
Curso Normal de Eventos	

Acción del Autor	Respuestas del Sistema
1. Este caso de uso empieza cuando el mensaje codificado es convertido en binario. 3. se separa en partes de 8 bits 4. se procede a emparejar. 5.se envían las parejas	**2.** Muestra el mensaje codificado

Tabla 3.6 Tabla de Caso de Uso de la Función Combinación

Caso de uso:	Combinacion
Actores:	Mensaje enviado/recibido
Propósito:	Combinar las parejas
Resumen:	Las parejas seleccionadas se combinan para formar un nuevo mensaje codificado
Tipo:	Primario
Pre condiciones:	Debe haber parejas seleccionadas
Curso Normal de Eventos	
Acción del Autor	**Respuestas del Sistema**
1. Este caso de uso empieza cuando se ha seleccionado las parejas. 3. se combinan las parejas 5. Se envía el nuevo mensaje.	**2.** Muestra el mensaje codificado 4. Muestra el mensaje combinado

Tabla 3.7 Tabla de Caso de Uso de la Función Mutación

Caso de uso:	Mutacion
Actores:	Mensaje enviado/recibido
Propósito:	Mutar el nuevo mensaje
Resumen:	El nuevo mensaje es cambiado uno de sus valores aleatorio
Tipo:	Primario
Pre condiciones:	Nuevo mensaje
Curso Normal de Eventos	

Acción del Autor	Respuestas del Sistema
1. Este caso de uso empieza cuando se tiene el nuevo mensaje 3. se altera uno de sus valores aleatoriamente 5. Se envía el nuevo mensaje.	**2.** Muestra el mensaje codificado 4. Muestra el mensaje combinado

Flujo Principal

1. Emisor: Redacta un mensaje
2. Sistema: El mensaje es codificado
3. Sistema: Al mensaje codificado se le aplica algoritmo genético
4. Sistema: El nuevo mensaje es enviado por cualquier canal de comunicación.
5. Sistema: El mensaje es recibido
6. Sistema: Se procede a aplicar algoritmo genético.
7. Sistema: El mensaje se decodificado.
8. Receptor: El mensaje es leído

3.3.4.1.3 Diagrama de estado

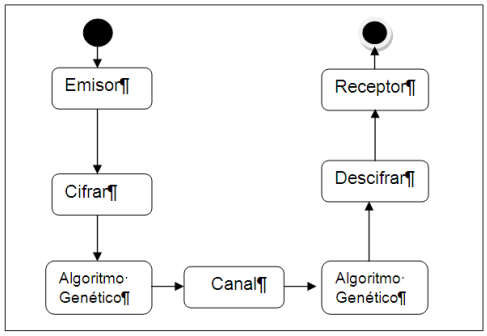

Figura 3.17: Diagrama de Estado

(Fuente Propia)

3.3.4.1.4 Diagrama de Bloques

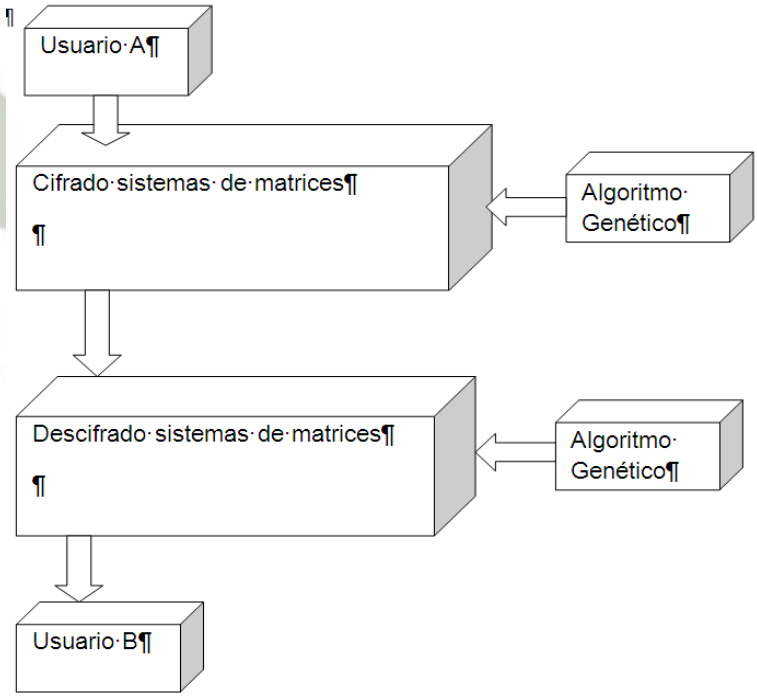

Figura 3.18: Diagrama de Bloques

(Fuente Propia)

3.3.4.1.5 Diagrama de Despliegue

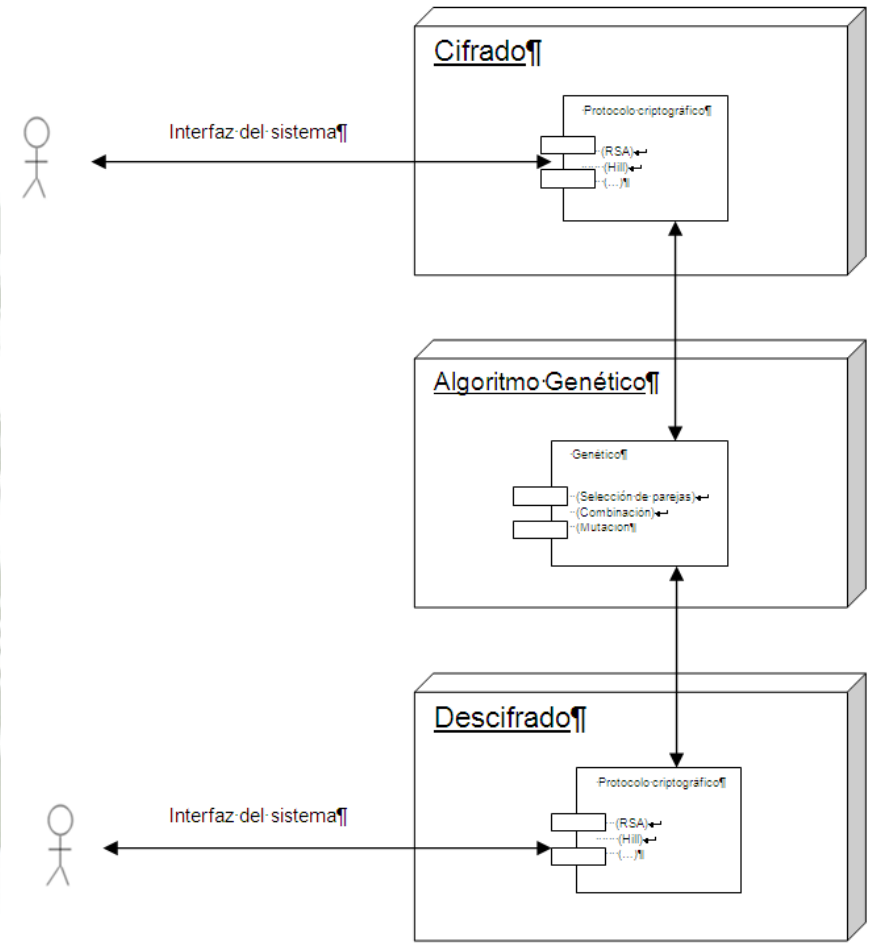

Figura 3.19: Diagrama de Despliegue

(Fuente Propia)

3.3.4.1.6 Diagrama de componentes

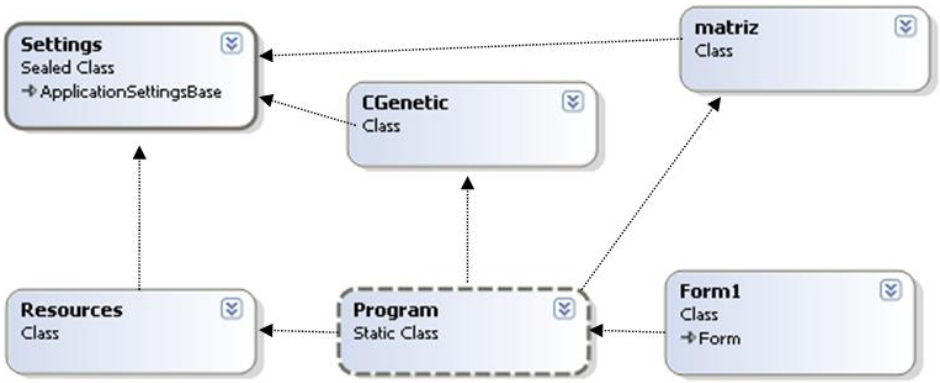

Figura 3.20: Diagrama de Componentes

(Fuente Propia)

Aplicando las mejoras y modificaciones se procedio a realizar un cambio a la interfas del prototipo para su prueba e implementacion para los usuarios.

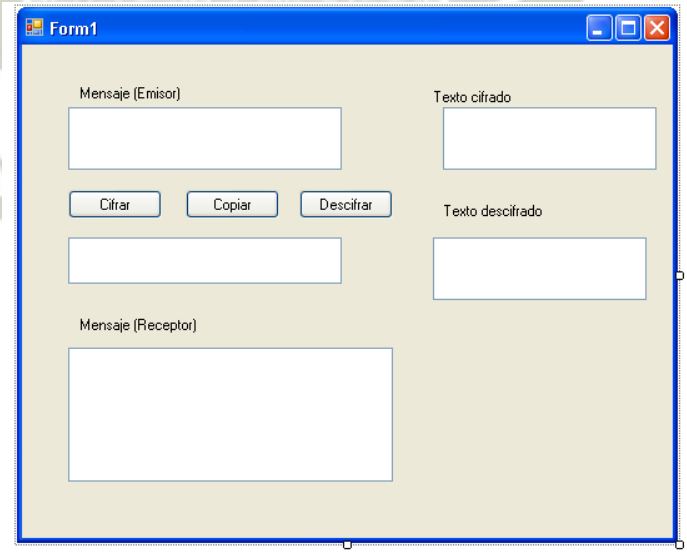

Figura 3.21: Interfaz del sistema

(Fuente Propia)

3.3.4.2 CODIFICACION

Tomando como base el prototipo anterios se añadira la codificacion del algoritmo genetico pero antes de su codificacion primero se vera un poco de los metodos a utilizar.

3.3.4.2.1 IDEA DEL ALGORITMO

Un Algoritmo Genético es una técnica de búsqueda basada en la teoría de la evolución de Darwin, en el cual los individuos más aptos de una población son los que sobreviven, al adaptarse más fácilmente a los cambios que se producen en su entorno

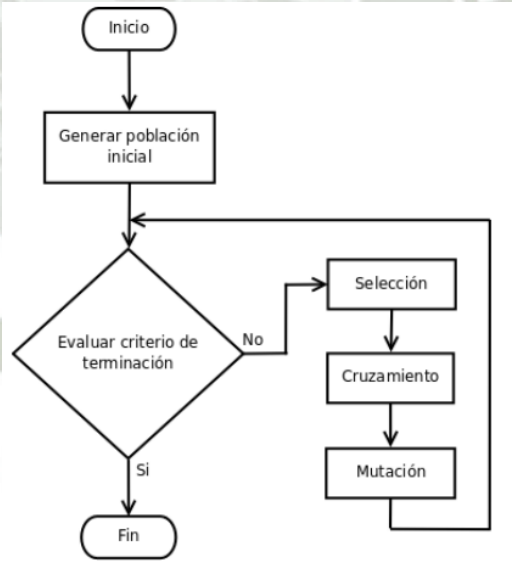

Figura 3.22: Diagrama de Flujo del Algoritmo Genético

(Haupt & Haupt, 2004)

3.3.4.2.1.1 Selección

Uno de los operadores básicos de un AG es el de selección, este esta basado en el concepto Darwiniano de “supervivencia del más apto”. Hay dos clases de métodos de selección usados comúnmente: los proporcionales y los basados en el orden . La selección proporcional, selecciona a los individuos basándose en el peso de su aptitud con respecto a los demás individuos. La selección basada en el orden, en cambio, crea una lista de individuos, ordenada de manera descendente por su aptitud, y selecciona los individuos teniendo en cuenta su posición en la lista.

- Selección por Torneo: en la selección por torneo se seleccionan al azar s individuos de la población, y se los compara para elegir al más apto. Este proceso se repite hasta seleccionar la cantidad de individuos deseada. Este método es apropiado principalmente cuando hay poblaciones grandes, donde por ejemplo, al utilizar selección por ranking, el ordenamiento de todos los individuos puede consumir mucho tiempo computacional.

3.3.4.2.1.2 Reproducción

Una vez seleccionados los individuos más aptos, se procede a la reproducción de estos. Se podría decir

que el operador de cruza, es el operador principal y que distingue a los AGs. La idea del cruzamiento es generar nuevos individuos a partir de la combinación genética de los padres. A continuación se describen algunos de los mecanismos de cruza más conocidos:

- Cruza Simple: si cada cromosoma tiene una longitud N, se selecciona al azar una posición de corte c, tal que 1 <= c < N. Luego se intercambian las partes de cada padre separadas por esa posición, generando así dos hijos. La Figura muestra un ejemplo de este método de cruza:

Figura 3.23: Ejemplo de Cruce Simple

(Fuente Propia)

3.3.4.2.1.3 Mutación

El operador de mutación es usado por los AGs para diversificar los cromosomas con el objeto de explotar el espacio de búsqueda lo máximo posible. Básicamente, consiste en una alteración aleatoria de los valores de los genes de cada cromosoma. La probabilidad de mutación, Pm, es usualmente pequeña

(1 en 1000). A causa de su baja probabilidad de ocurrencia, la mutación se considera un operador secundario de los AGs. La probabilidad de mutación puede ser constante durante toda la búsqueda (Mutación simple), como así también adaptarse según el grado de convergencia (Mutación Adaptativa por Convergencia) de la búsqueda o por la cantidad de generaciones (Mutación Adaptativa por Temperatura). La mutación simple se ocupa usualmente debido a su sencillez.

3.3.4.2.2 Código fuente Algoritmo Genético

```
classCGenetic
{


staticintcolumnas = 2;
staticintNganadores = 3;
staticdoublesumatoria = 0;



intpuntocruce;
int gen;


string[,] poblacion;
string[,] poblaciontem;
string[] parejas;
string[] ganadores;



publicCGenetic(intNfilas, intPuntoC, int Gen)
      {


string[,] poblacion = newstring[Nfilas,columnas] ;
string[,] poblaciontem = newstring[Nfilas, columnas];
string[] parejas = newstring[Nfilas];
string[] ganadores = newstring[Nganadores];


this.poblacion = poblacion;
this.poblaciontem = poblaciontem;
this.parejas = parejas;
```

```
this.ganadores = ganadores;
this.puntocruce = PuntoC;
this.gen = Gen;
        }



publicstring[,] AlgoritmoGenetico(char[] pob, intNfilas)
{

iniciarpoblacion(pob, Nfilas);

seleccion_parejas(poblacion, parejas);
combinacion_mutacion(poblacion, poblaciontem, parejas);

returnpoblacion;
        }

publicvoidiniciarpoblacion(char[] pob,int count)
        {
string j = string.Empty;
for (inti = 0; i< count; i++)
            {
                j = ToBinary(pob[i]);
poblacion[i, 1] = j;
poblacion[i, 0] = "" + i;

            }

        }


publicvoidcombinacion_mutacion(string[,] poblacion, string[,] poblaciontem,
string[] parejas)
        {
string[] individuoA;
string[] individuoB;
string[] NewCadenaA;
string[] NewCadenaB;
stringparejaA = "";

for (inti = 0; i<parejas.Length / 2; i++)
            {
parejaA = parejas[i];
individuoA = newstring[poblacion[i, 1].Length];
individuoB = newstring[poblacion[int.Parse(parejaA), 1].Length];
string[] cadenaA = poblacion[i, 1].Split();
string[] cadenaB = poblacion[int.Parse(parejaA), 1].Split();
stringcadAdnA = string.Join("", cadenaA);
stringcadAdnB = string.Join("", cadenaB);


for (int j = 0; j <individuoA.Length; j++)
                {
string A = cadAdnA.Substring(j, 1);
string B = cadAdnB.Substring(j, 1);
individuoA[j] = A;
individuoB[j] = B;
}
```

```
NewCadenaA = newstring[individuoA.Length];
NewCadenaB = newstring[individuoB.Length];

for (int t = 0; t <puntocruce; t++)
            {
NewCadenaA[t] = individuoA[t];
NewCadenaB[t] = individuoB[t];

            }

for (int t = puntocruce; t <individuoA.Length; t++)
            {
NewCadenaA[t] = individuoB[t];
NewCadenaB[t] = individuoA[t];

            }

stringcadAdnAB = string.Join("", NewCadenaA);
poblaciontem[i, 0] = "" + i;
poblaciontem[i, 1] = cadAdnAB;

stringcadAdnBA = string.Join("", NewCadenaB);
poblaciontem[(parejas.Length-1)-i, 1] = cadAdnBA;
}
for (inti = 0; i<parejas.Length; i++)
{
poblacion[i, 0] = poblaciontem[i, 0];
poblacion[i, 1] = poblaciontem[i, 1];
}

string Cad;
string[] Mutado;
for (inti = 0; i<parejas.Length; i++)
        {
Mutado = newstring[poblacion[i, 1].Length];
string[] cadena = poblacion[i, 1].Split();
            Cad = string.Join("", cadena);

for (int j = 0; j <Mutado.Length; j++)
            {
string A = Cad.Substring(j, 1);
Mutado[j] = A;
            }

Cad = "";
if (Mutado[gen].Equals("0")) { Mutado[gen] = "1"; }
else { Mutado[gen] = "0"; }
Cad = string.Join("", Mutado);
poblacion[i, 1] = Cad;

}
      }


publicvoidseleccion_parejas(string[,] poblacion, string[] parejas)
      {
string aux = poblacion[1, 0];
for (inti = 0; i<parejas.Length; i++)
{
```

```
parejas[(parejas.Length - 1) - i] = poblacion[i, 0];
}
      }


publicvoidadaptabilidad(string[,] poblacion, doublesumatoria, intNfilas)
{
for (inti = 0; i<Nfilas; i++)
{
poblacion[i, 4] = "" + (double.Parse(poblacion[i, 3]) / sumatoria);
}
      }


publicstaticvoidverpoblacion(string[,] poblacion, Booleanpareja, intNfilas,
string[] parejas)
{
string cadena="";

for(inti=0;i<Nfilas;i++)
{
for(int k=0;k<columnas;k++)
{
                  cadena+="["+poblacion[i,k]+"]";
            }
if(pareja){cadena+="pareja"+parejas[i]+"\n";}
else{cadena+="\n";}
         }
Console.WriteLine(cadena);
}

publicdoublefuncion_fx(double X)
      {
return (X*X);
      }

publicstaticstringToBinary(charstr)
      {
string converted = string.Empty;
byte[] encoding = Encoding.ASCII.GetBytes(str.ToString());

for (inti = 0; i<encoding.Length; i++)
         {
for (int j = 0; j < 8; j++)
            {
converted += (encoding[i] & 0x80) > 0 ? "1" :"0";
encoding[i] <<= 1;
            }
         }

return converted;

      }


publicstaticByte[] GetBytesFromBinaryString(string binary)
      {
var list = newList<Byte>();
```

```
for (inti = 0; i<binary.Length; i += 8)
        {
string t = binary.Substring(i, 8);

list.Add(Convert.ToByte(t, 2));
}

returnlist.ToArray();
      }
```

CAPITULO IV:

PRUEBAS REALIZADAS Y RESULTADOS

4.1 GENERALIDADES

Concluido el estudio en esta parte se presentan los resultados obtenidos en la encuesta realizada a 25 usuarios, a través del Facebook y a 25 usuarios de una empresa, así como los resultados obtenidos en las pruebas realizadas al software.

Se realizó utilizando el muestreo por cotas, ya que los encuestados si desean contestan a los preguntas formuladas.

Se describe los resultados estadísticos e interpretativos en cuadros y gráficos en barras cuyos datos han sido obtenidos mediante la aplicación de la encuesta a los usuarios hackers y crackers a través de internet.

Así mismo, se describe las conclusiones y finalmente las sugerencias que estoy seguro servirá de guía e iniciativa para estudios similares o más profundos

4.2 PRUEBAS

A continuación se muestra cuadros de resúmenes de las pruebas realizadas al software, en estos cuadros mostramos la eficiencia y la eficacia en el cual le tomo al software encriptar y desencriptar la información.

Tabla 4.1Tamaño y Tiempo

Tamaño	T (encriptar)	T(desencriptar)
100 KB	1,5 seg	1,5 seg
500 KB	2 seg	2,5 seg
1 MB	3 seg	3,5 seg

Análisis e Interpretación:

En el presente cuadro se observa a mayor tamaño del mensaje más es el tiempo que le toma al sistema en encriptarlo, también indicamos que para una mejor utilización del sistema se recomendaría enviar mensajes con un tamaño menor o igual a 1MB

Tabla 4.2Eficiencia

Ejecución del sistema	**F (encriptar) %**	**F (desencriptar) %**
10 veces	100%	100%
20 veces	100%	100%
50 veces	100%	100%

Análisis e Interpretación:

En la siguiente tabla se muestra que el sistema se ejecutó más de 50 veces, indicando que todas las veces que el sistema se ejecutó tubo un 100% en poder encripta con éxito los mensajes enviados, al momento de

desencriptar se muestra que a las 10 veces de ser ejecutado el sistema tubo un 100% en poder descifrar el mensaje encriptado, a las 20 veces el sistema tubo el mismo resultado y a las 50 veces de ejecutado el sistema tubo también u porcentaje del 100% en poder descifrar el mensaje. Esto nos indica que el sistema puede cifrar cualquier mensaje en un 100% y puede descifrar el mismo mensaje en un 100%

Tabla 4.3 Mensaje Desencriptado es Igual al Mensaje Original

Alternativas	Veces ejecutada	%
a) Si	30	100
b) No	0	0
Total	30	100

Análisis e Interpretación:

El siguiente cuadro indica si el mensaje descifrado es igual al mensaje original, para tal propósito el sistema se ejecutó 30 veces en las cuales las 30 veces el mensaje descifrado es igual al mensaje original.

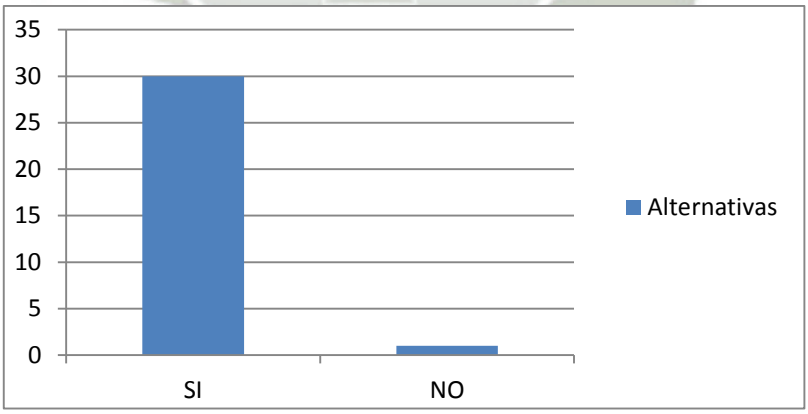

Grafico 4.1: Grafico de la Tabla 4.3

(Fuente Propia)

4.3 ENCUESTAS REALIZADAS

Después de la aplicación de la encuesta a los usuarios, se obtuvieron resultados significativos que permiten comprobar la mejora en la seguridad que se tiene en la información privilegiada al usar el software.

Los cuadros y gráficas en barras que a continuación se expone son los resultados de interpretación de los datos reportados por el propio estudio

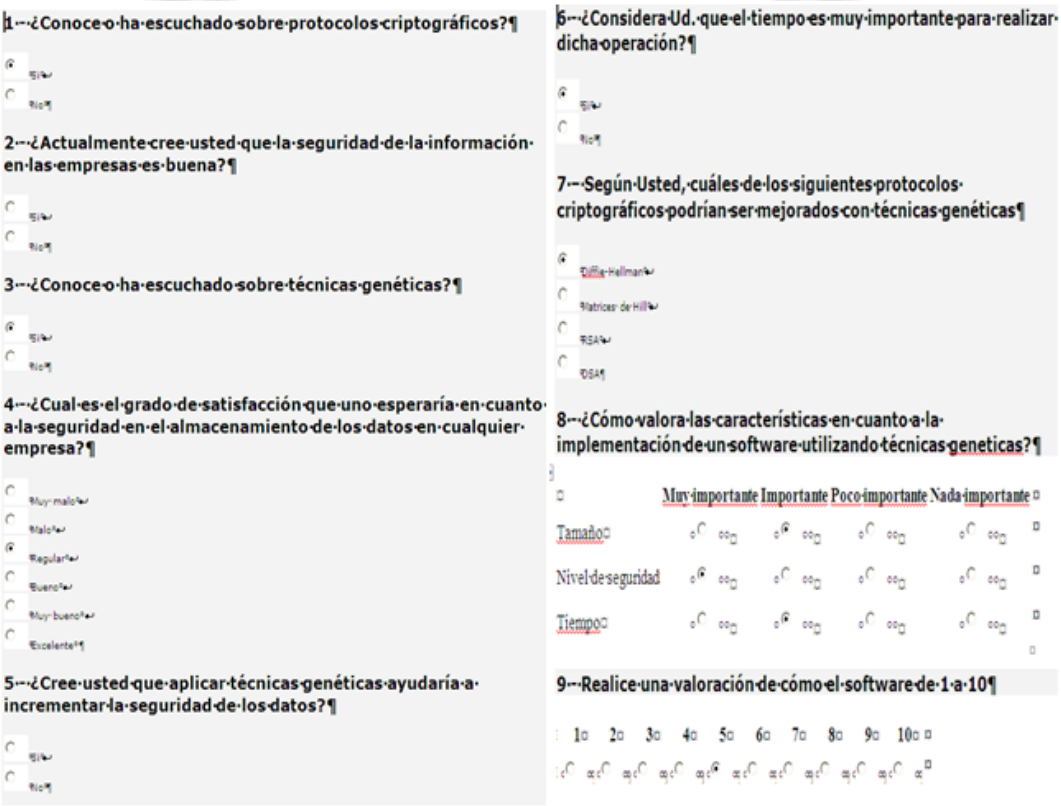

Figura 4.1: Encuesta

(Fuente Propia)

Tabla 4.4 Interés por Descifrar el Mensaje

Alternativas	f	%
a) Mucho	17	68,0
b) Poco	6	24,0
c) Nada	2	8,0
Total	25	100,0

Análisis e Interpretación:

En el presente cuadro se observa que 17 (68%) usuarios encuestados, respondieron que les interesa mucho descifrar el mensaje encriptado que se les proporcionó, 6 (24%) usuarios, indicaron que es poco el interés que tienen, mientras que sólo 2 (8%) usuarios indicaron que no tienen interés por descifrar el mensaje encriptado.

De la información significativa obtenida se deduce que la mayoría, representado por 17 (68%) usuarios, tienen mucho interés por descifrar el mensaje encriptado, ya que la motivación por el hecho de hacerles suponer que era una información muy privilegiada, o simplemente por una naturaleza propia humana el hecho de querer saber aquello que se encuentra oculto

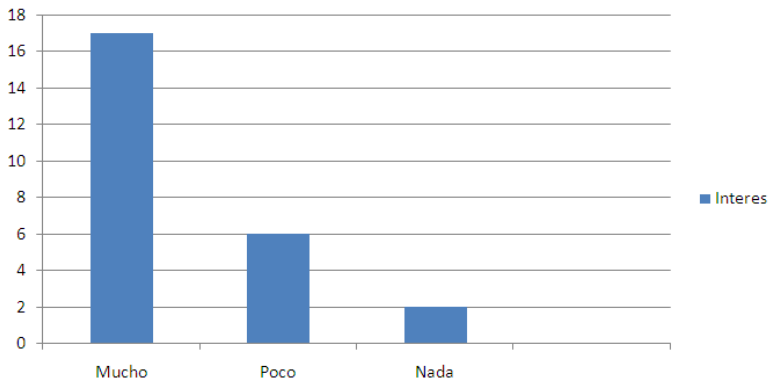

Grafico 4.2: Grafico de la Tabla 4.4

(Fuente Propia)

Tabla 4.5 Técnicas Genéticas Ayudan a Incrementar la Seguridad de los Datos

Alternativas	f	%
a) SI	38	68,0
b) NO	5	24,0
c) NINGUNO	2	8,0
Total	45	100,0

Análisis e Interpretación:

En el presente cuadro se observa que 38 (68%) usuarios encuestados, respondieron que la seguridad de los datos incrementaría al utilizar técnicas genéticas, 5 (24%) usuarios, indicaron que aun utilizando técnicas genéticas los datos seguirían inseguros, mientras que sólo 2 (8%) usuarios indicaron no tener importancia ya que no conocen que son técnicas genéticas.

De la información significativa obtenida se deduce que la mayoría, representado por 38 (68%) usuarios, tienen mucho interés en la seguridad de dato, lo cual implica que al utilizar técnicas genéticas brindan una capa más de seguridad a los datos.

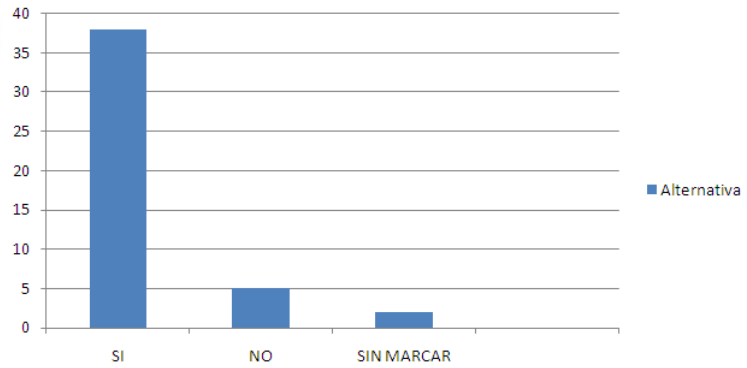

Grafico 4.3: Grafico de la Tabla 4.5

(Fuente Propia)

Tabla 4.6 Tamaño en la Implementación del Software Utilizando Tecnicas Geneticas

Alternativas	f	%
a) Muy Importante	5	8
b) Importante	35	84
c) Poco Importante	5	8
d) Nada Importante	0	0
Total	45	100,0

Análisis e Interpretación:

En el presente cuadro se observa que 35 (88%) usuarios encuestados, respondieron que el tamaño de los datos era importante al implementar el software, 5 (8%) usuarios, indicaron que era muy importante el tamaño de datos para el software utilizando técnicas genéticas, otros 8 (8%) usuarios indicaron que era poco importante el tamaño de los datos.

Analizando el cuadro se llega a la conclusión que el tamaño de los datos juega un papel importante al desarrollar el software.

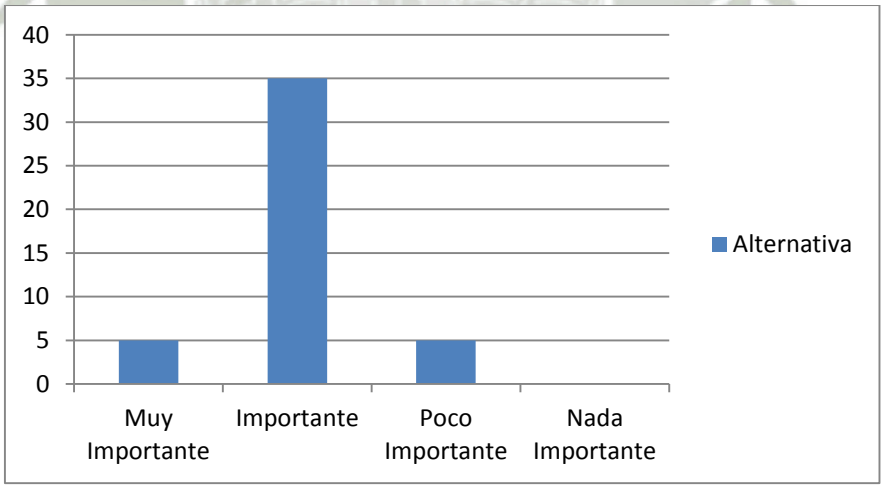

Grafico 4.4: Grafico de la Tabla 4.6

(Fuente Propia)

Tabla 4.7 Nivel de Seguridad en la Implementación del Software Utilizando Técnicas Genéticas

Alternativas	f	%
a) Muy Importante	40	90
b) Importante	5	10
c) Poco Importante	0	0
d) Nada Importante	0	0
Total	45	100,0

Análisis e Interpretación:

En el presente cuadro se observa que 40 (90%) usuarios encuestados, respondieron que la seguridad de los datos son muy importantes al implementar el software, 5 (8%) usuarios, indicaron que era importante la seguridad de datos para el software utilizando técnicas genéticas.

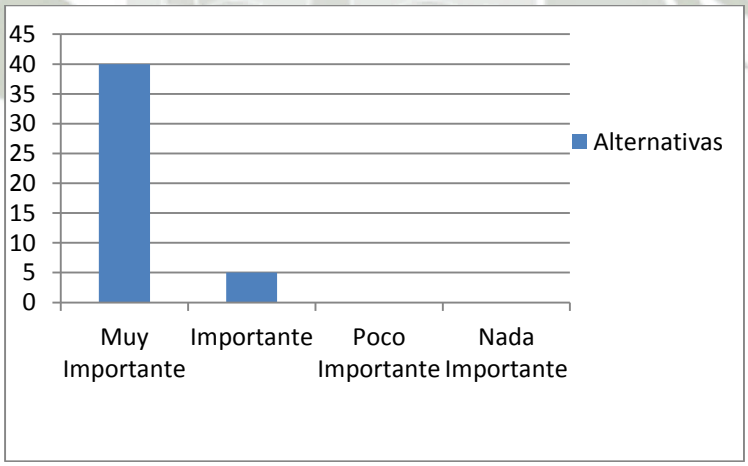

Grafico 4.5: Grafico de la Tabla 4.7

(Fuente Propia)

Tabla 4.8 Tiempo en la Implementación del Software Utilizando Técnicas Genéticas

Alternativas	f	%
a) Muy Importante	5	10
b) Importante	40	90
c) Poco Importante	0	0
d) Nada Importante	0	0
Total	45	100,0

Análisis e Interpretación:

En el presente cuadro se observa que 5 (10%) usuarios encuestados, respondieron que el tiempo de encriptación y des encriptado de los datos era importante al implementar el software, 40 (90%) usuarios, indicaron que era muy importante el tiempo de encriptación y des encriptado de datos para el software utilizando técnicas genéticas,

Analizando el cuadro se llega a la conclusión que el tiempo de encriptación y des encriptado de los datos es importante al desarrollar el software.

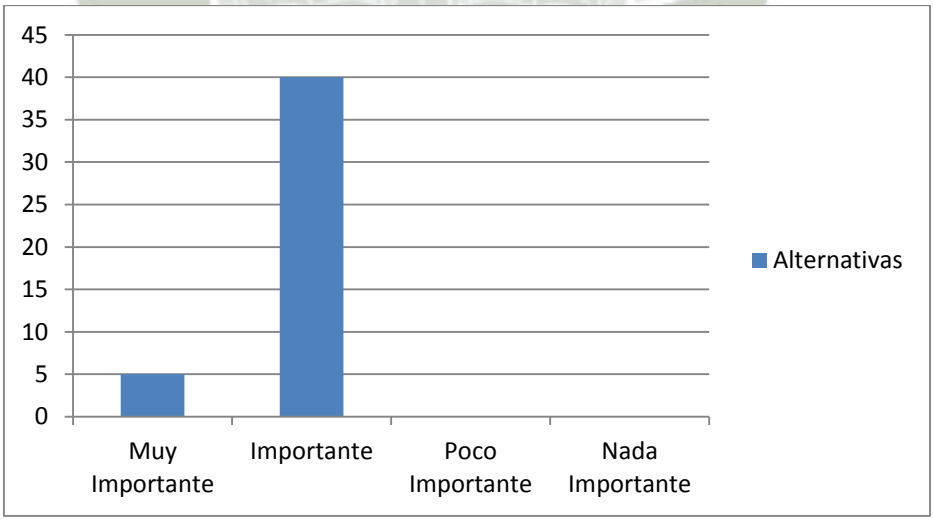

Grafico 4.6: Grafico de la Tabla 4.8

(Fuente Propia)

CONCLUSIONES

1. Los algoritmos evaluados aunque muestran que tienen vulnerabilidades no quiere decir que sean obsoletos ya que eso dependería mucho del tipo de aplicación y el nivel de seguridad que se requiera, pero mientras más moderno sea el algoritmo utilizado menos vulnerabilidades conocidas puede tener.

2. La eficiencia de los algoritmos genéticos depende de las nuevas generaciones que se obtienen, es decir que mientras se obtengan nuevas poblaciones de individuos se incrementara su eficiencia.

3. Al comparar el algoritmo desarrollado con respecto a los algoritmos evaluados podemos concluir que:

   - Tamaño: el tamaño de la información enviada no altera las funciones de los algoritmos genéticos (selección, cruce y mutación) pero si los ejecuta con mayor lentitud.

   - Tiempo: El tiempo en la encriptación y desencriptado de datos se incrementa ya que se ve influenciado con respecto al tamaño del mensaje, es decir, mientras más pesado el mensaje más es el tiempo que toma el software en encriptarlo y desencriptarlo.

   - Nivel de seguridad: El nivel de seguridad se ve incrementado al utilizar algoritmos genéticos, también se ve que se puede dar solución a algunas vulnerabilidades de algunos algoritmos criptográficos.

RECOMENDACIONES

1. Se realizara una nueva versión con los resultados obtenidos para brindar una mejor implementación en cuanto a la seguridad de los datos.

2. Al momento de ingresar a las funciones de los algoritmos genéticos se realizar más de una iteración con el motivo de incrementar la seguridad de datos.

3. Las próximas pruebas se realizaran en un centro de trabajo para ver el desempeño de los algoritmos genéticos implementando las nuevas actualizaciones.

BIBLIOGRAFIA

Borisov, N., Golberg, I., & Brewer, E. (2004). Off the Record Communication or, Why not to use PGP.

Cipher A., D., Kahn, D., Kruh, L., & Mellen, G. (01 de December de 1987). Cryptology: Yesterday, Today, and Tomorrow. Artech House Publishers.

Diffie, W., & Hellman, M. (1976). New Directions in cryptography, IEEE Transactions on Information Theory. págs. 644-654.

Gordon, D. (1993). Designing and Detecting Trapdoors for Discrete Log. Crytosystems Advances in Crytology. Springer Verlag.

Gutierrez Gutierrez, J. (2003). *Protocolos criptograficos y seguridad en redes.* universidad de cantabria.

Haupt, R. L., & Haupt, S. E. (2004). *Practical Genetic Algorithms.* Wiley.

Lipson, J. D. (1981). Elements of Algebra and Algebraic Computing. Addison-Wesley.

Lucena Lopez, M. J. (Junio de 2010). Criptografìa y Seguridad en Computadoras. Universidad de Jaèn.

Mitchell, M. (1999). An introduccion to Genetic Algorithms. MIT Press.

Newton, D. E. (01 de october de 1997). Encyclopedia of Crytology. *ABC-CLIO*.

Ramio Aguirre, J. (marzo de 2006). Seguridad Informatica. *Universidad Politecnica de Madrid*. Cuarta Edicion v32.